finalProject

December 14, 2022

[26]:

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

import pandas as pd
import numpy as np
import math
from plotnine import *

from sklearn import metrics
from sklearn.preprocessing import StandardScaler #Z-score variables

# For logistic regression
from sklearn.linear_model import LogisticRegression, Ridge, Lasso # Logistic␣
 ↪Regression Model
from sklearn.tree import DecisionTreeClassifier # Decision Tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
from sklearn.model_selection import cross_val_score # cross validation metrics
from sklearn.model_selection import cross_val_predict # cross validation metrics
from sklearn.model_selection import GridSearchCV
from sklearn.metrics._plot.roc_curve import plot_roc_curve

# Ridge/Lasso Models
from sklearn.linear_model import RidgeCV, LassoCV

# Clustering Model, Gaussian Mixture
from sklearn.mixture import GaussianMixture

from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score,␣
 ↪plot_confusion_matrix, f1_score, recall_score, precision_score
```

[27]:

```
fields = ['GenderSelect', 'Country', 'Age', 'EmploymentStatus', 'CodeWriter',␣
 ↪'StudentStatus', 'CurrentJobTitleSelect', 'LanguageRecommendationSelect',␣
 ↪'LearningDataScienceTime', 'TimeSpentStudying', 'FormalEducation',␣
 ↪'CompensationAmount', 'JobHuntTime', 'EmployerSearchMethod',␣
 ↪'WorkToolsSelect']
df = pd.read_csv("Datasets/multipleChoiceResponses.csv", usecols = fields ,␣
 ↪encoding = 'latin-1')

df.head()
```

[27]:

```
                                        GenderSelect        Country   Age  \
0  Non-binary, genderqueer, or gender non-conforming            NaN   NaN
1                                             Female  United States  30.0
2                                               Male         Canada  28.0
3                                               Male  United States  56.0
4                                               Male         Taiwan  38.0

                                    EmploymentStatus StudentStatus CodeWriter  \
0                                 Employed full-time           NaN        Yes
1                 Not employed, but looking for work           NaN        NaN
2                 Not employed, but looking for work           NaN        NaN
3  Independent contractor, freelancer, or self-em…           NaN        Yes
4                                 Employed full-time           NaN        Yes

              CurrentJobTitleSelect LanguageRecommendationSelect  \
0             DBA/Database Engineer                           F#
1                               NaN                       Python
2                               NaN                            R
3  Operations Research Practitioner                       Python
4                Computer Scientist                       Python

  LearningDataScienceTime TimeSpentStudying    FormalEducation  \
0                     NaN               NaN  Bachelor's degree
1               1-2 years      2 - 10 hours    Master's degree
2               1-2 years      2 - 10 hours    Master's degree
3                     NaN               NaN    Master's degree
4                     NaN               NaN    Doctoral degree

                                EmployerSearchMethod  \
0  I visited the company's Web site and found a j…
1                                                NaN
2                                                NaN
3                                                NaN
4                          A tech-specific job board

                                     WorkToolsSelect CompensationAmount  \
0  Amazon Web services,Oracle Data Mining/ Oracle…                NaN
```

```
1                                                NaN                NaN
2                                                NaN                NaN
3  Amazon Machine Learning,Amazon Web services,Cl…           250,000
4  C/C++,Jupyter notebooks,MATLAB/Octave,Python,R…               NaN

  JobHuntTime
0         NaN
1         NaN
2         1-2
3         NaN
4         NaN
```

[28]:

```python
df.shape
```

[28]:

```
(16716, 15)
```

[29]:

```python
# Cleaning up the data

# Simplify data in GenderSelect
df = df[df['GenderSelect'].isin(['Male','Female'])]

# Only United States Participants
df = df[df['Country'] == 'United States']

# Drop those who didnt input their age
df = df.dropna(subset=['Age'])

# Drop 'prefer not to answer' for FormalEducation
df = df[~df['FormalEducation'].isin(['I prefer not to answer'])]

# Drop 'Some other way' for EmployerSearchMethod
df = df[~df['EmployerSearchMethod'].isin(['Some other way'])]

# Simplify EmploymentStatus, if employed: 1, else: 0
df['isEmployed'] = df['EmploymentStatus'].apply(lambda x: 1 if any(s in x for s in ['full-time', 'freelancer']) else 0)

# Convert StudentStatus null values to 0
df['StudentStatus'] = df['StudentStatus'].astype(str).apply(lambda x: 1 if 'Yes' in x else 0)

# Only show those who have a job in analyzing data, software or programming
jobTypes = ['Data', 'Software', 'Computer', 'Database', 'Business Analyst', 'Machine Learning', 'Programmer']
df['CodeWriter'] = df['CodeWriter'].apply(lambda x: 1 if x == 'Yes' else 0)
df = df[df['CurrentJobTitleSelect'].isin(jobTypes) | df['CodeWriter'] == 1]
```

```
# Drop those who didnt put their annual salary
df = df.dropna(subset=['CompensationAmount'])

# Convert the salary from a string to an integer value
df['CompensationAmount'] = df['CompensationAmount'].apply(lambda x: int(float(x.
 ↪replace(',', ''))))
```

[30]:

```
# Simplifying labels
df['FormalEducation'] = df['FormalEducation'].replace('I did not complete any␣
 ↪formal education past high school', 'high_school')
df['FormalEducation'] = df['FormalEducation'].replace('Some college/university␣
 ↪study without earning a bachelor\'s degree', 'some_college')
df['FormalEducation'] = df['FormalEducation'].replace('Professional degree',␣
 ↪'professional')
df['FormalEducation'] = df['FormalEducation'].replace('Bachelor\'s degree',␣
 ↪'bachelors')
df['FormalEducation'] = df['FormalEducation'].replace('Master\'s degree',␣
 ↪'masters')
df['FormalEducation'] = df['FormalEducation'].replace('Doctoral degree', 'PhD')

df['EmployerSearchMethod'] = df['EmployerSearchMethod'].replace('A career fair␣
 ↪or on-campus recruiting event', 'campus_recruitment')
df['EmployerSearchMethod'] = df['EmployerSearchMethod'].replace('A friend,␣
 ↪family member, or former colleague told me', 'referral')
df['EmployerSearchMethod'] = df['EmployerSearchMethod'].replace(['A␣
 ↪general-purpose job board', 'A tech-specific job board', 'I visited the␣
 ↪company\'s Web site and found a job listing there'], 'job_board')
df['EmployerSearchMethod'] = df['EmployerSearchMethod'].replace(['I was␣
 ↪contacted directly by someone at the company (e.g. internal recruiter)', 'An␣
 ↪external recruiter or headhunter'], 'recruiter')
```

[31]:

```
# Create dummies for GenderSelect, FormalEducation and EmployerSearchMethod
dummies = pd.get_dummies(df['GenderSelect'])
df = pd.concat([df, dummies], axis = 1)
df = df.drop('GenderSelect', 1)

dummies = pd.get_dummies(df['FormalEducation'])
df = pd.concat([df, dummies], axis = 1)
df = df.drop('FormalEducation', 1)

dummies = pd.get_dummies(df['EmployerSearchMethod'])
df = pd.concat([df, dummies], axis = 1)
df = df.drop('EmployerSearchMethod', 1)
```

[32]:

```
df.head()
```

[32]:

```
          Country   Age                                   EmploymentStatus  \
3   United States  56.0  Independent contractor, freelancer, or self-em…
15  United States  58.0  Independent contractor, freelancer, or self-em…
22  United States  33.0                                 Employed full-time
34  United States  35.0                                 Employed full-time
75  United States  40.0                                 Employed full-time

    StudentStatus  CodeWriter              CurrentJobTitleSelect  \
3               0           1  Operations Research Practitioner
15              0           1             DBA/Database Engineer
22              0           1              Scientist/Researcher
34              0           1                          Engineer
75              0           1              Scientist/Researcher

   LanguageRecommendationSelect LearningDataScienceTime TimeSpentStudying  \
3                        Python                     NaN               NaN
15                            R                     NaN               NaN
22                       Matlab                     NaN               NaN
34                       Python                     NaN               NaN
75                       Python                     NaN               NaN

                                      WorkToolsSelect  …  PhD bachelors  \
3   Amazon Machine Learning,Amazon Web services,Cl…  …    0         0
15  C/C++,IBM Cognos,MATLAB/Octave,Microsoft Excel…  …    0         0
22                               MATLAB/Octave,Python  …    1         0
34  MATLAB/Octave,Python,R,SAS JMP,SQL,TIBCO Spotfire  …    1         0
75              Amazon Machine Learning,C/C++,NoSQL,R  …    1         0

    high_school  masters  professional  some_college  campus_recruitment  \
3             0        1             0             0                   0
15            0        1             0             0                   0
22            0        0             0             0                   0
34            0        0             0             0                   0
75            0        0             0             0                   0

    job_board  recruiter  referral
3           0          0         0
15          0          0         0
22          0          0         1
34          0          0         1
75          0          0         1

[5 rows x 25 columns]
```

[33]:

```
predictors = ['Male','Female','Age', 'CompensationAmount',
    'PhD', 'bachelors', 'high_school', 'masters', 'professional',
    'some_college', 'campus_recruitment', 'job_board', 'recruiter',
    'referral']
continuous_variables = ['Age', 'CompensationAmount']

X = df[predictors]
y = df['isEmployed']
```

[34]:

```
#80/10 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# zscore (only continous and interval variables)
z = StandardScaler()
X_train[continuous_variables] = z.fit_transform(X_train[continuous_variables])
X_test[continuous_variables] = z.transform(X_test[continuous_variables])
```

[35]:

```
# Logistic Regression Model
lr = LogisticRegression()
lr.fit(X_train, y_train) # test set should never see the inside of a .fit
```

[35]:

```
LogisticRegression()
```

[36]:

```
# Performance Metrics
print("Train Acc: ", accuracy_score(y_train, lr.predict(X_train)))
print("Test Acc: ", accuracy_score(y_test, lr.predict(X_test)))

print("TRAIN Precision: ", precision_score(y_train, lr.predict(X_train)))
print("TEST Precision : ", precision_score(y_test, lr.predict(X_test)))

print("TRAIN Recall: ", recall_score(y_train, lr.predict(X_train)))
print("TEST Recall : ", recall_score(y_test, lr.predict(X_test)))

print("TRAIN ROC/AUC: ", roc_auc_score(y_train, lr.predict_proba(X_train)[:,1]))
print("TEST ROC/AUC : ", roc_auc_score(y_test, lr.predict_proba(X_test)[:,1]))
```

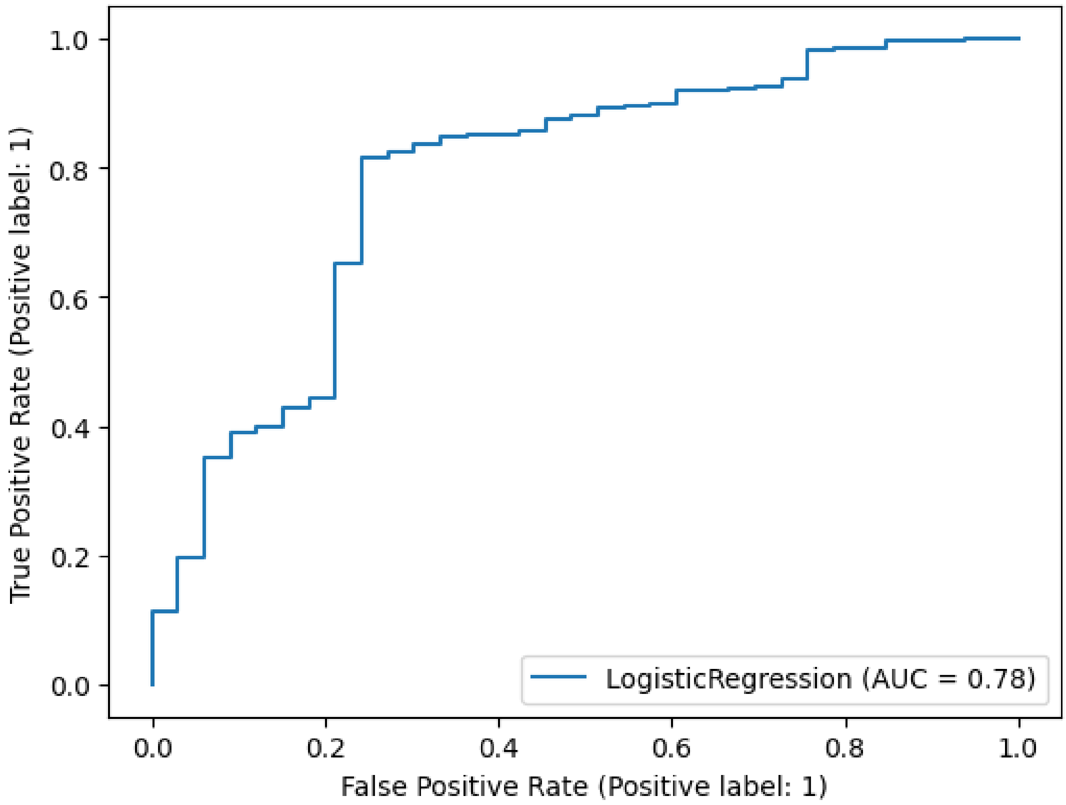
```
Train Acc:  0.9689849624060151
Test Acc:  0.951310861423221
TRAIN Precision:  0.9689849624060151
TEST Precision :  0.951310861423221
TRAIN Recall:  1.0
TEST Recall :  1.0
TRAIN ROC/AUC:  0.7835875731123064
TEST ROC/AUC :  0.6025136281041793
```

[37]:

```
# ROC Curve for LR
plot_roc_curve(lr, X_train, y_train)
```

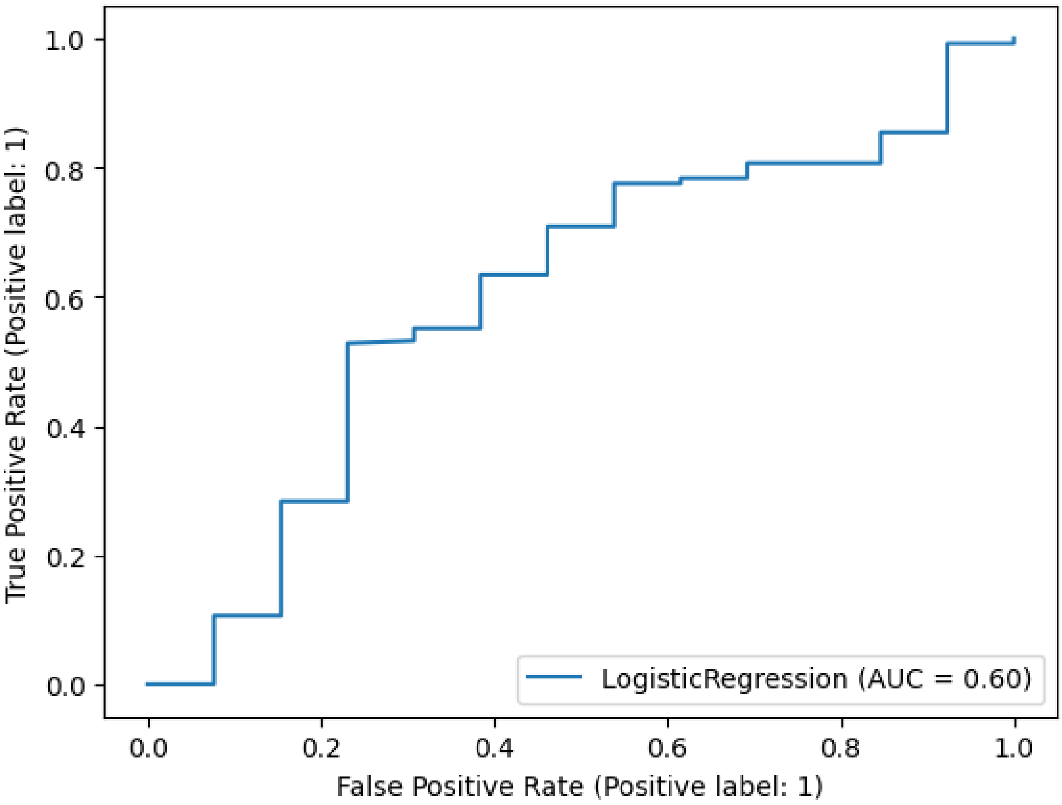
[37]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x106cc0b80>
```



[38]:
```
# ROC Curve for LR
plot_roc_curve(lr, X_test, y_test)
```

[38]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x2887d45b0>
```



```
[39]: lasso_model = Lasso()
      lasso_model.fit(X_train, y_train)

      print("TRAIN: ", mean_absolute_error(y_train, lasso_model.predict(X_train)))
      print("TEST : ", mean_absolute_error(y_test, lasso_model.predict(X_test)))
```

```
TRAIN:  0.060106210074057295
TEST :  0.07668398524400886
```

```
[40]: ridge_model = Ridge()
      ridge_model.fit(X_train, y_train)

      print("TRAIN: ", mean_absolute_error(y_train, ridge_model.predict(X_train)))
      print("TEST : ", mean_absolute_error(y_test, ridge_model.predict(X_test)))
```

```
TRAIN:  0.06380915063680255
TEST :  0.081029587436048
```

```
[41]: # LASSO
      predictors = ['Male','Female','Age', 'CompensationAmount',
          'PhD', 'bachelors', 'high_school', 'masters', 'professional',
          'some_college', 'campus_recruitment', 'job_board', 'recruiter',
          'referral']
      continuous_variables = ['Age', 'CompensationAmount']

      X = df[predictors]
      y = df['isEmployed']

      #80/10 split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

      # zscore (only continous and interval variables)
      z = StandardScaler()
      X_train[continuous_variables] = z.fit_transform(X_train[continuous_variables])
      X_test[continuous_variables] = z.transform(X_test[continuous_variables])

      lsr_tune = LassoCV(cv = 5).fit(X_train,y_train)
      # lsr_tune = LassoCV(cv = 5, alphas = [0.001,0.01,0.05,1]).fit(X_train,y_train)

      print("TRAIN: ", mean_absolute_error(y_train, lsr_tune.predict(X_train)))
      print("TEST : ", mean_absolute_error(y_test, lsr_tune.predict(X_test)))

      print("\nwe chose " + str(lsr_tune.alpha_) + " as our alpha.")
```

```
TRAIN:  0.07477043489143065
TEST :  124.41368238267599

we chose 0.0010694794140082694 as our alpha.
```

[42]:

```
# Ridge tuning
predictors = ['Male','Female','Age', 'CompensationAmount',
    'PhD', 'bachelors', 'high_school', 'masters', 'professional',
    'some_college', 'campus_recruitment', 'job_board', 'recruiter',
    'referral']
continuous_variables = ['Age', 'CompensationAmount']

X = df[predictors]
y = df['isEmployed']

#80/10 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# zscore (only continous and interval variables)
z = StandardScaler()
X_train[continuous_variables] = z.fit_transform(X_train[continuous_variables])
X_test[continuous_variables] = z.transform(X_test[continuous_variables])

rr_tune = RidgeCV(cv = 5).fit(X_train,y_train)
```

```
print("TRAIN: ", mean_absolute_error(y_train, rr_tune.predict(X_train)))
print("TEST : ", mean_absolute_error(y_test, rr_tune.predict(X_test)))

print("\nwe chose " + str(rr_tune.alpha_) + " as our alpha.")
```

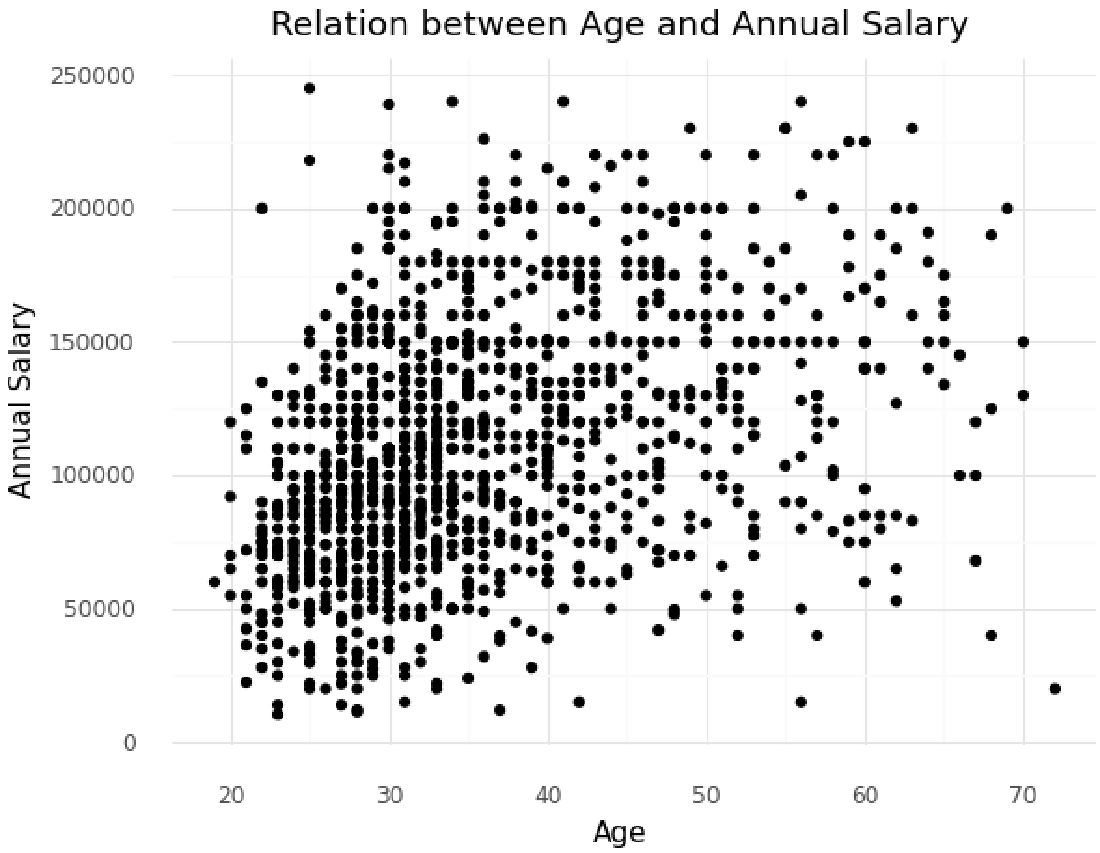
```
TRAIN:  0.0582307193038051
TEST :  0.08546631957736815

we chose 10.0 as our alpha.
```

[43]:

```
#from scipy.stats import zscore
#z_scores = zscore(df['CompensationAmount'])
#df_clean = df[(z_scores < 3).all(axis=1) & (z_scores > -3).all(axis=1)]

# Remove Outliers
df = df[df['CompensationAmount'] < 250_000]
df = df[df['CompensationAmount'] > 10_000]

df = df[df['Age'] > 18]
```

[44]:

```
(ggplot(df, aes(x = "Age", y = "CompensationAmount"))
    + geom_point()
    + theme_minimal()
    + ggtitle("Relation between Age and Annual Salary")
    + labs(x = "Age", y = "Annual Salary")
)
```

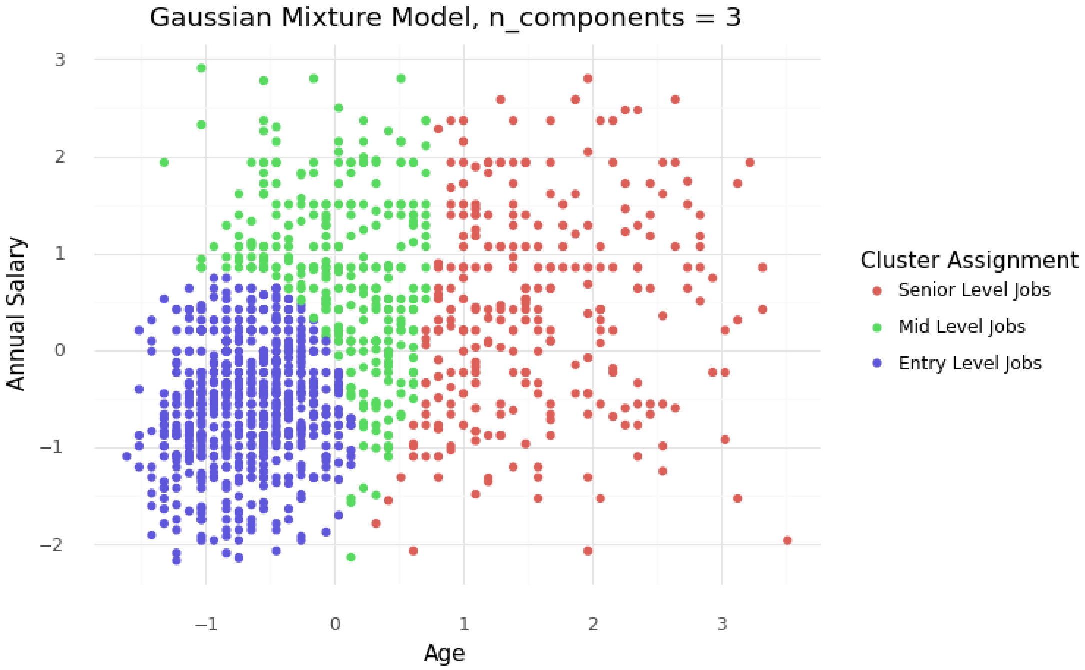


```
[44]: <ggplot: (679856171)>
```

```
[45]: from sklearn.preprocessing import StandardScaler

      features = ['Age', 'CompensationAmount']
      z = StandardScaler()

      df[features] = z.fit_transform(df[features])
      gmm = GaussianMixture(n_components = 3).fit(df[features])
      gmm_labels = gmm.predict(df[features])

      #labList = ["Cluster " + str(i) for i in range(1, len(set(gmm.labels_)))]

      df['assignments'] = gmm_labels
```

```
[49]: (ggplot(df, aes(x = "Age", y = "CompensationAmount", color =␣
       ↪"factor(assignments)"))
          + geom_point()
          + theme_minimal()
```

```
    + labs(title = "Gaussian Mixture Model, n_components = 3", x = "Age", y = "Annual Salary")
    + scale_color_discrete(name = "Cluster Assignment", labels = (["Senior Level Jobs", "Mid Level Jobs", "Entry Level Jobs"]))
)
```



[49]: <ggplot: (680356255)>

[47]:
```
ss_gmm = silhouette_score(df[features], df[['assignments']])
print("Silhouette Score for GM Model:", ss_gmm)
```

```
Silhouette Score for GM Model: 0.34963370059537546
```