finalProject

December 15, 2022

1 CPSC 392 Final Project, Fall 2022

1.0.1 Gilberto Arellano | Student ID: 1801074

Im attempting to create predictive models to figure out what makes a succesful candidate to land a job in tech. I will be using data from the 2017 Kaggle survey, in which collected responses in regards to people's salary, education, programming language recommendations, etc.

1.0.2 Questions to Answer

- 1) Who is the most ideal candidate to land a job in tech? What are the characteristics needed to land a job in this field? Are employers targeting people with a college education? Did an internship in undergrad? A couple of portfolios under their belt? Or is having a solid reference enough? We will use a logistic regression model since the output we are predicting is what are the variables needed to land a job. Doing so we can analyze the coefficients and decide which factors has the most impact
- 2) Are employers targeting people through job boards or referals? We are going to use both the logistic regression and regularaztization methods to see where employers are mostly looking for candidates through.
- 3) Is there a relation between age and wage? After plotting the relationship between 'Age' and 'SalaryCompensation', I determined to use Gaussian Mixture as the clustering method since there is no clear seperation in which other models such as DBSCAN would struggle to differentiate.

[14]:
```python
import warnings
warnings.filterwarnings('ignore')

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

# Pre-processing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Models
from sklearn.linear_model import LinearRegression, LogisticRegression, Ridge,␣
↪Lasso
```

```
from sklearn.linear_model import RidgeCV, LassoCV
from sklearn.mixture import GaussianMixture

# Metrics
from sklearn.metrics import *
from sklearn.metrics._plot.roc_curve import plot_roc_curve
```

[15]:
```
# Importing data
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()
```

[15]:
```
                                       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
```

1.1 Data Cleanup

[16]:
```python
# 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 = df[~df['EmploymentStatus'].isin(['I prefer not to say'])]
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'].map(lambda x: 1 if x == "Yes" else 0)
df['StudentStatus'] = df['StudentStatus'].astype(str).apply(lambda x: 1 if␣
↪"Yes" in x else 0)
```

```
# To keep the students, conver their salary to 0. Prevents their rows being␣
↪dropped when using df.dropna()
selected_rows = df.loc[df["StudentStatus"] == 1]
df.loc[selected_rows.index, "CompensationAmount"] = 0

# If the value is a string, remove their comma and convert to an integer
df = df.dropna(subset=['CompensationAmount']) # Drop any responses that didnt␣
↪include their salary
df['CompensationAmount'] = df['CompensationAmount'].apply(lambda x: int(float(x.
↪replace(',', '')) if isinstance(x, str) else x))
```

[17]:
```
# 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_degree')
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')
```

[18]:
```
# 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)
```

[19]:
```
df.head()
```

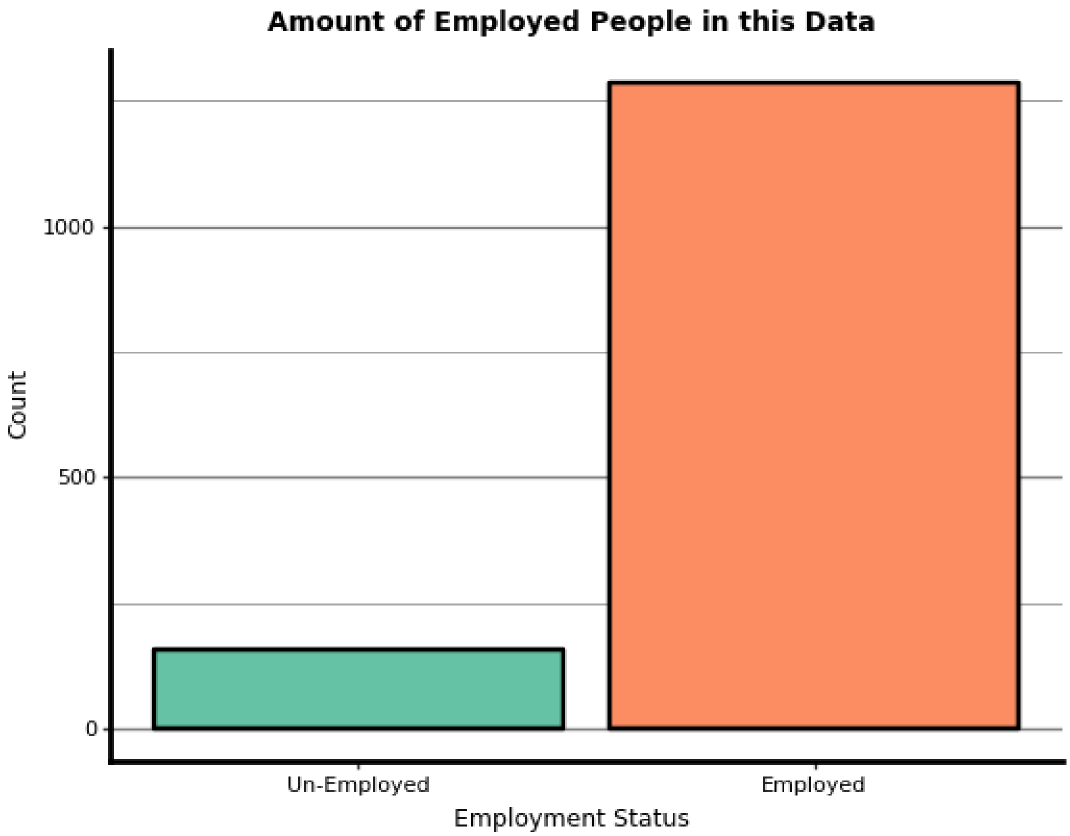
[19]:
```
         Country   Age                                EmploymentStatus  \
3   United States  56.0  Independent contractor, freelancer, or self-em…
15  United States  58.0  Independent contractor, freelancer, or self-em…
22  United States  33.0                               Employed full-time
34  United States  35.0                               Employed full-time
43  United States  22.0           Not employed, and not looking for work

    StudentStatus CodeWriter             CurrentJobTitleSelect  \
3               0        Yes  Operations Research Practitioner
15              0        Yes             DBA/Database Engineer
22              0        Yes              Scientist/Researcher
34              0        Yes                          Engineer
43              1        NaN                               NaN

   LanguageRecommendationSelect LearningDataScienceTime TimeSpentStudying  \
3                        Python                     NaN               NaN
15                            R                     NaN               NaN
22                       Matlab                     NaN               NaN
34                       Python                     NaN               NaN
43                            R                < 1 year      2 - 10 hours

                                        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
43                                                   NaN  …    0          0

    high_school  masters  professional_degree  some_college  \
3             0        1                    0             0
15            0        1                    0             0
22            0        0                    0             0
34            0        0                    0             0
43            0        0                    0             1

    campus_recruitment  job_board  recruiter  referral
3                    0          0          0         0
15                   0          0          0         0
22                   0          0          0         1
34                   0          0          0         1
43                   0          0          0         0
```

```
[5 rows x 25 columns]
```

1.2 Information Regarding the Data

After cleaning up the data and filtering the responses to be from the US only. The majority in this dataset are employed, meaning our logistic regression model can have high a high accuracy rate at predicting employment and possibly skewing the results. Low amount of students responded to this survey.

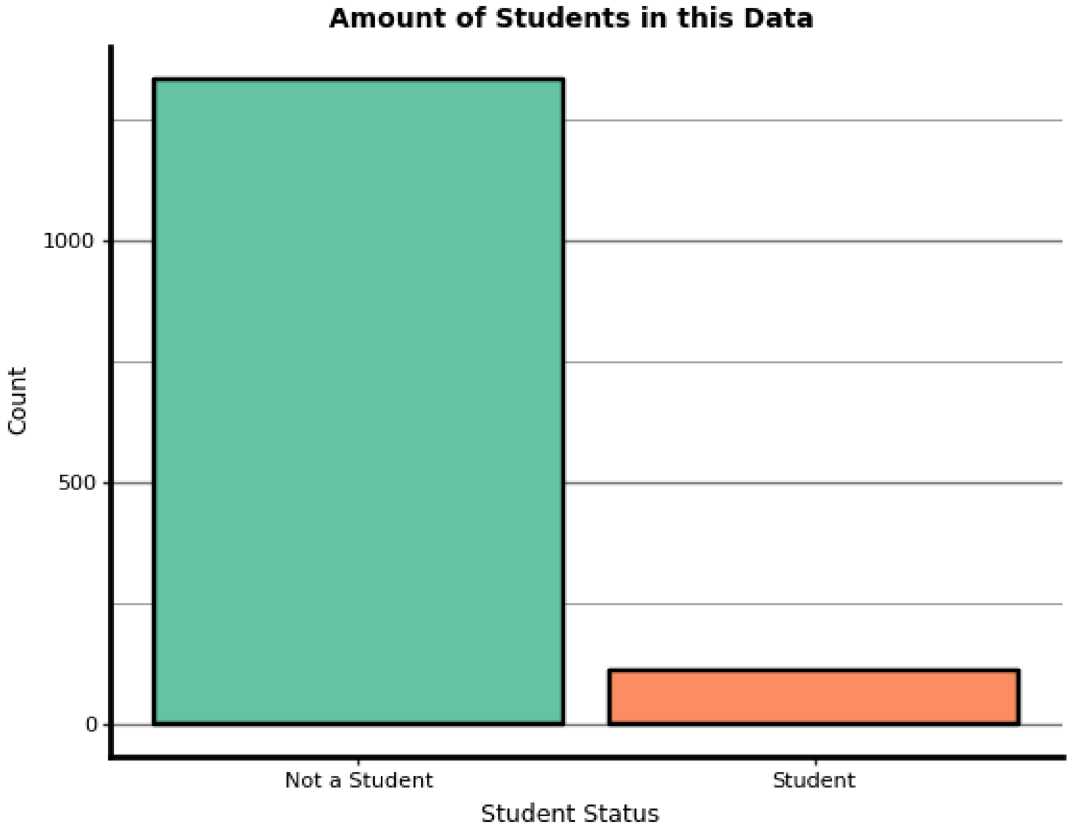
[20]:
```
(ggplot(df, aes(x = "isEmployed", fill = "factor(isEmployed)"))
    + geom_bar(color = "black", size = 1)
    + ggtitle("Amount of Employed People in this Data")
    + labs(x = "Employment Status", y = "Count")
    + scale_x_continuous(breaks = (0,1), labels = ['Un-Employed', 'Employed'])
    + scale_fill_brewer(type ='qual', palette='Set2')
    + theme( # Customizes grid
        axis_line = element_line(size = 2, color = "black"),
        panel_border = element_blank(),
        panel_background = element_blank(),
        panel_grid_major_y = element_line(color = "gray", size = 1),
        panel_grid_minor_y = element_line(color = "gray", size = 0.5),
    )
    + theme( # Customizes text
        plot_title = element_text(size = 10, face = "bold"),
        text = element_text(size = 9),
        axis_text_x = element_text(color="black", size = 8),
        axis_text_y = element_text(color="black", size = 8),
        legend_position = "none",
))
```



[20]: <ggplot: (705959126)>

[21]:
```
(ggplot(df, aes(x = "StudentStatus", fill = "factor(StudentStatus)"))
    + geom_bar(color = "black", size = 1)
    + ggtitle("Amount of Students in this Data")
    + labs(x = "Student Status", y = "Count")
    + scale_x_continuous(breaks = (0,1), labels = ['Not a Student', 'Student'])
    + scale_fill_brewer(type ='qual', palette='Set2')
    + theme( # Customizes grid
        axis_line = element_line(size = 2, color = "black"),
        panel_border = element_blank(),
        panel_background = element_blank(),
        panel_grid_major_y = element_line(color = "gray", size = 1),
        panel_grid_minor_y = element_line(color = "gray", size = 0.5),
    )
    + theme( # Customizes text
        plot_title = element_text(size = 10, face = "bold"),
        text = element_text(size = 9),
        axis_text_x = element_text(color="black", size = 8),
        axis_text_y = element_text(color="black", size = 8),
```

```
        legend_position = "none",
))
```



[21]: <ggplot: (705925217)>

1.3 Analyzing the Data

[22]:
```
# Preparing variables for supervised models
predictors = ['Male','Female','Age',
    'PhD', 'bachelors', 'high_school', 'masters', 'professional_degree',
    'some_college', 'campus_recruitment', 'job_board', 'recruiter',
    'referral']
continuous_variables = ['Age']

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

[23]:
```
#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])
```

1.3.1 Logistic Regression

[24]:
```
# Logistic Regression Model
lr = LogisticRegression()
lr.fit(X_train, y_train) # test set should never see the inside of a .fit
```

[24]: LogisticRegression()

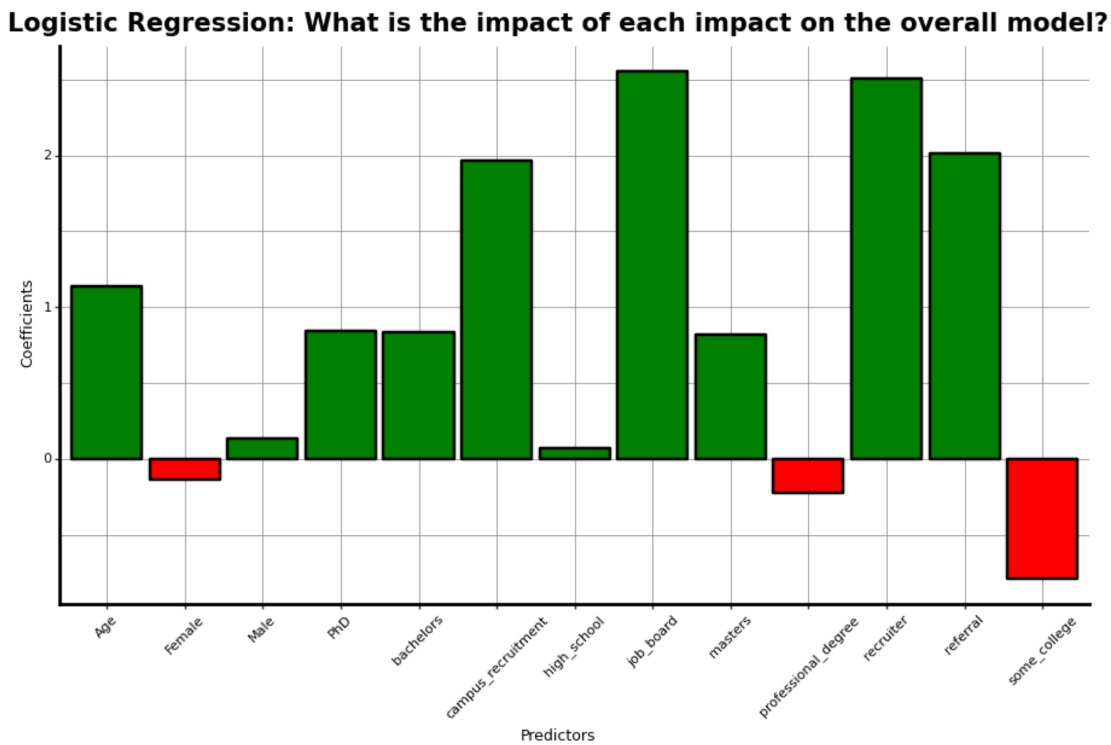
[25]:
```
# Performance Metrics
print("Train Acc: ", accuracy_score(y_train, lr.predict(X_train)))
print("Test Acc: ", accuracy_score(y_test, lr.predict(X_test)))

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]))

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

```
Train Acc:  0.9322916666666666
Test Acc:  0.9480968858131488
TRAIN Precision:  0.93646408839779
TEST Precision :  0.9518518518518518
TRAIN Recall:  0.9912280701754386
TEST Recall :  0.9922779922779923
TRAIN ROC/AUC:  0.8831376280206689
TEST ROC/AUC :  0.9337837837837837
TRAIN MAE:  0.06770833333333333
TEST MAE:  0.05190311418685121
```

[26]:
```
# Coefficient values from the LR model
coefficients = pd.DataFrame({
    "Coefficients": lr.coef_[0],
    "Name": predictors
})

# Create a dictionary to easily identify positive and negative values
```

```
color_values = {}
for coef, name in coefficients[["Coefficients", "Name"]].values:
    if coef > 0:
        color_values[str(name)] = "green"
    else:
        color_values[str(name)] = "red"
```

[27]:
```
(ggplot(coefficients, aes(x = "Name", y = "Coefficients", fill =␣
 ↪"factor(Name)"))
    + scale_fill_manual(values = color_values)
    + geom_bar(stat = "identity", position = "identity", color = "black", size␣
 ↪= 1)
    + labs(x = "Predictors", y = "Coefficients")
    + ggtitle("Logistic Regression: What is the impact of each impact on the␣
 ↪overall model?")
    + labs(x = "Predictors")
    + theme( # Customizes grid
        axis_line = element_line(size = 2, color = "black"),
        panel_border = element_blank(),
        panel_background = element_blank(),
        panel_grid_major_y = element_line(color = "gray", size = 0.5),
        panel_grid_minor_y = element_line(color = "gray", size = 0.5),
        panel_grid_major_x = element_line(color = "gray", size = 0.5),
    )
    + theme( # Customizes text
        plot_title = element_text(size = 15, face = "bold"),
        text = element_text(size = 9),
        axis_text_x = element_text(color="black", size = 8),
        axis_text_y = element_text(color="black", size = 8),
        legend_position = "none",
        )
    + theme(
        figure_size = (11, 6),
        axis_text_x = element_text(rotation=45))
)
```



[27]: <ggplot: (706031394)>

[28]:
```
# ROC Curve for LR
plot_roc_curve(lr, X_train, y_train)
```

[28]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x2a15ac1f0>

[29]:
```
# ROC Curve for LR
plot_roc_curve(lr, X_test, y_test)
```

[29]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x2a254c6a0>

1.3.2 Lasso Model

[30]:
```
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:  0.19482421875
TEST :  0.19047361591695502
```

[31]:

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

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:  0.14734751454059813
TEST :  0.13723151864621022

we chose 9.662738151838982e-05 as our alpha.
```

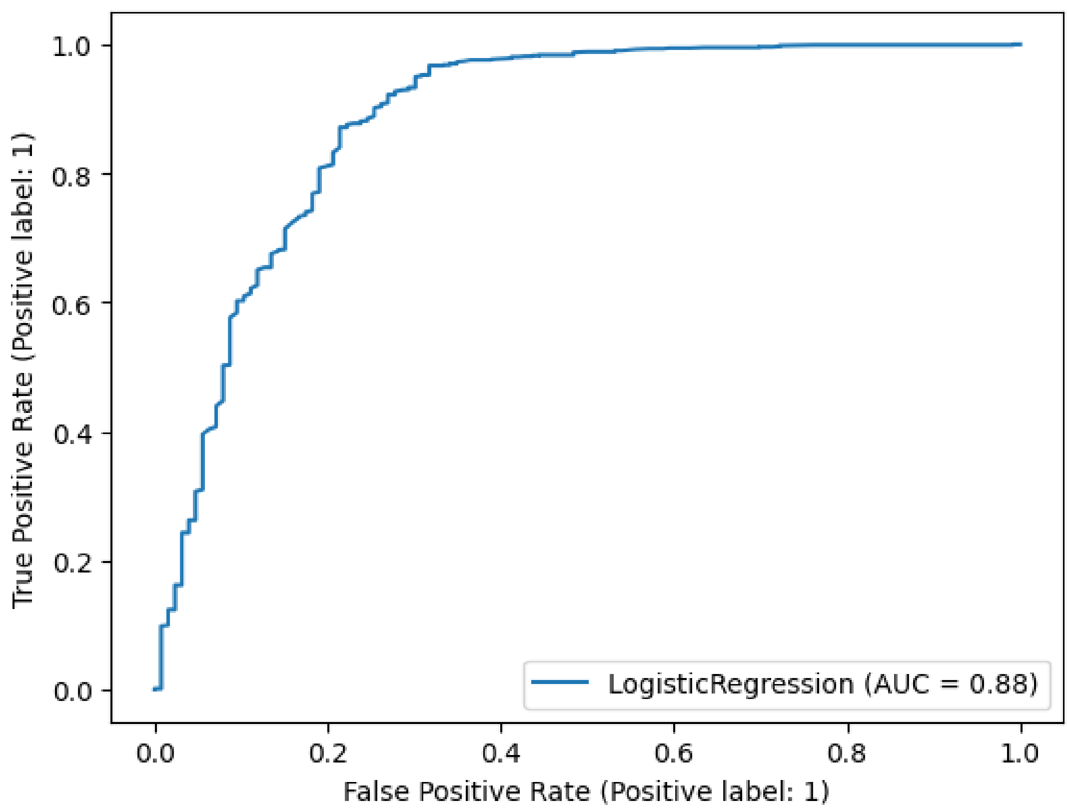
[32]:

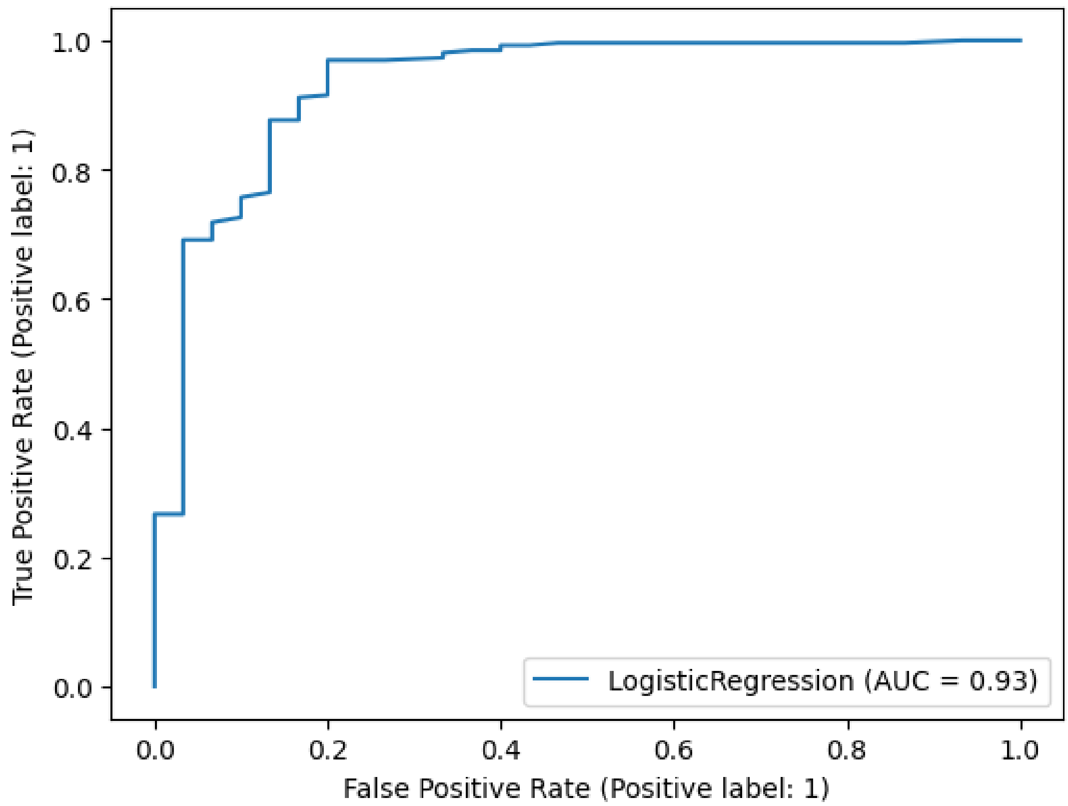
```
# Coefficient values from the LR model
lasso_coefficients = pd.DataFrame({
    "Coefficients": lsr_tune.coef_,
    "Name": predictors
})
```

[33]:

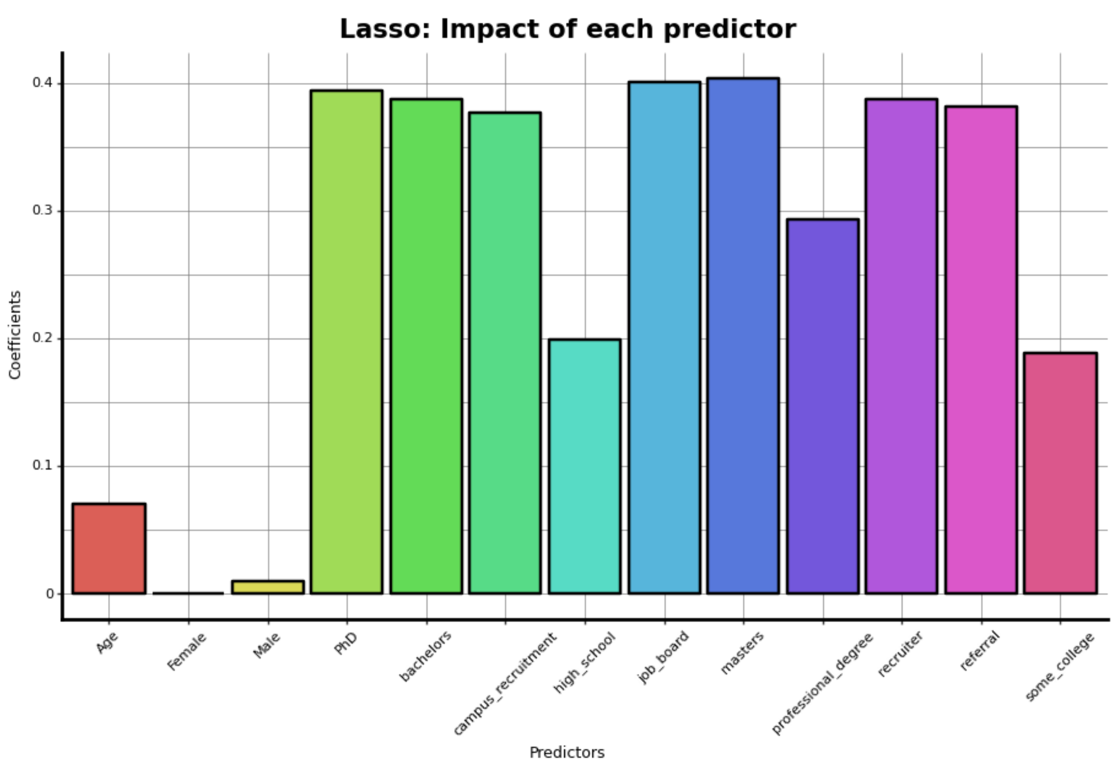
```
(ggplot(lasso_coefficients, aes(x = "Name", y = "Coefficients", fill =␣
 ↪"factor(Name)"))
    + geom_bar(stat = "identity", position = "identity", color = "black", size␣
 ↪= 1)
    + labs(x = "Predictors", y = "Coefficients")
    + ggtitle("Lasso: Impact of each predictor")
    + labs(x = "Predictors")
    + theme( # Customizes grid
        axis_line = element_line(size = 2, color = "black"),
        panel_border = element_blank(),
```

```
        panel_background = element_blank(),
        panel_grid_major_y = element_line(color = "gray", size = 0.5),
        panel_grid_minor_y = element_line(color = "gray", size = 0.5),
        panel_grid_major_x = element_line(color = "gray", size = 0.5),
    )
    + theme( # Customizes text
        plot_title = element_text(size = 15, face = "bold"),
        text = element_text(size = 9),
        axis_text_x = element_text(color="black", size = 8),
        axis_text_y = element_text(color="black", size = 8),
        legend_position = "none",
        )
    + theme(
        figure_size = (11, 6),
        axis_text_x = element_text(rotation=45))
)
```





LogisticRegression (AUC = 0.93)
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0



Lasso: Impact of each predictor
Coefficients
Predictors
Age
Female
Male
PhD
bachelors
campus_recruitment
high_school
job_board
masters
professional_degree
recruiter
referral
some_college
0
0.1
0.2
0.3
0.4

[33]: <ggplot: (707112894)>

1.3.3 Ridge Model

[34]:
```
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:  0.14866753341223612
TEST :  0.13804174265586158
```

[35]:
```
# Ridge tuning
predictors = ['Male','Female','Age',
    'PhD', 'bachelors', 'high_school', 'masters', 'professional_degree',
    'some_college', 'campus_recruitment', 'job_board', 'recruiter',
    'referral']
continuous_variables = ['Age']

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.")
```

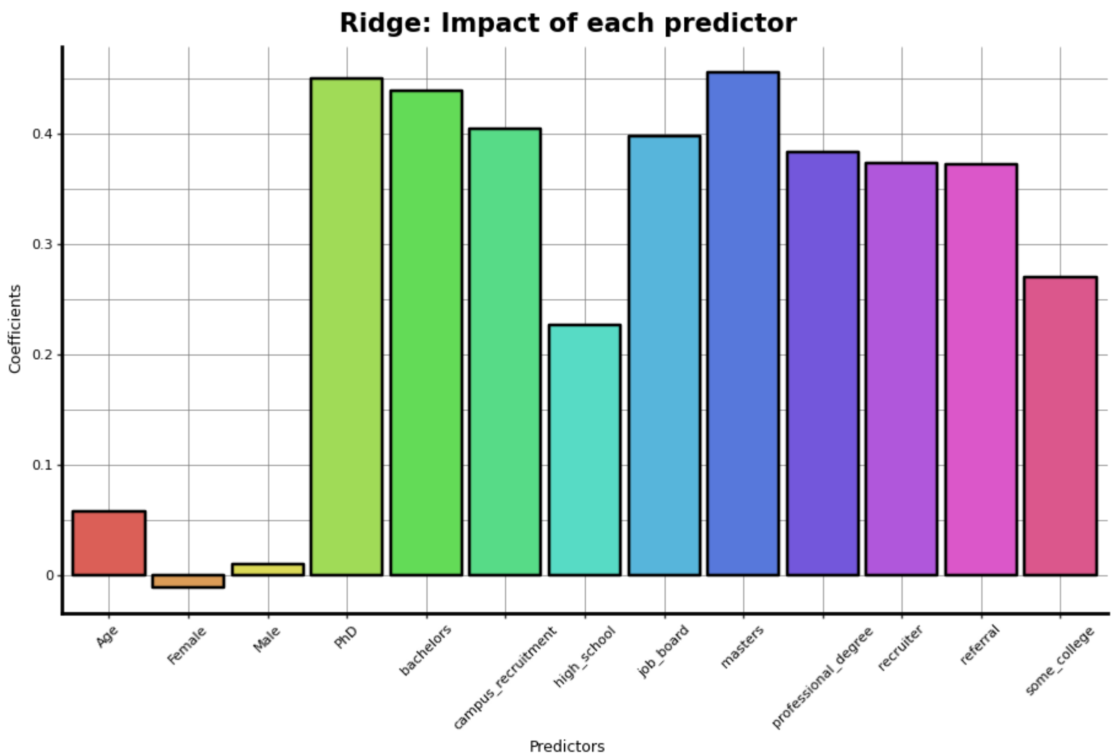
```
TRAIN:  0.1395771129244935
TEST :  0.13803578700165933

we chose 0.1 as our alpha.
```

[36]:

```
# Coefficient values from the Ridge Model
ridge_coefficients = pd.DataFrame({
    "Coefficients": rr_tune.coef_,
    "Name": predictors
})
```

[37]:

```
(ggplot(ridge_coefficients, aes(x = "Name", y = "Coefficients", fill =␣
↪"factor(Name)"))
    + geom_bar(stat = "identity", position = "identity", color = "black", size␣
↪= 1)
    + labs(x = "Predictors", y = "Coefficients")
    + ggtitle("Ridge: Impact of each predictor")
    + labs(x = "Predictors")
    + theme( # Customizes grid
        axis_line = element_line(size = 2, color = "black"),
        panel_border = element_blank(),
        panel_background = element_blank(),
        panel_grid_major_y = element_line(color = "gray", size = 0.5),
        panel_grid_minor_y = element_line(color = "gray", size = 0.5),
        panel_grid_major_x = element_line(color = "gray", size = 0.5),
    )
    + theme( # Customizes text
        plot_title = element_text(size = 15, face = "bold"),
        text = element_text(size = 9),
        axis_text_x = element_text(color="black", size = 8),
        axis_text_y = element_text(color="black", size = 8),
        legend_position = "none",
        )
    + theme(
        figure_size = (11, 6),
        axis_text_x = element_text(rotation=45))
)
```



[37]: <ggplot: (707071133)>

1.3.4 Guassian Mixture Clustering

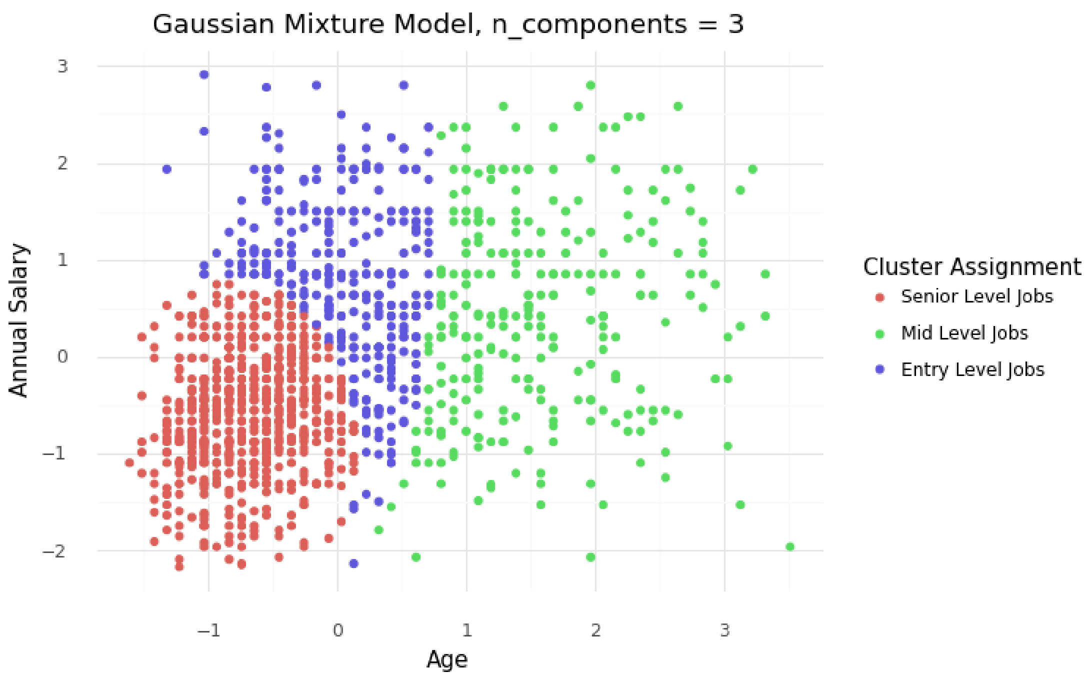
[38]:
```
df = df[df['CompensationAmount'] < 250_000]
df = df[df['CompensationAmount'] > 10_000]
df = df[df['Age'] > 18]

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

[38]: <ggplot: (709345612)>

[39]:
```
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
```

[40]:
```
(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"]))
    + theme_minimal()
)
```



[40]: <ggplot: (709408919)>

[41]:
```
ss_gmm = silhouette_score(df[features], df[['assignments']])
print("Silhouette Score for GM Model:", ss_gmm)
```

```
Silhouette Score for GM Model: 0.34994320117656524
```

1.3.5 Results

Despite our logistic regression model having a high accuracy rate, it is skewed since the majority of the responses are from people who are employed. We do not have sufficient data to characterize what could possible be an unemployed person.

Our Logistic and Lasso model performed similar with a MAE at around 0.19. Although we are going to use the Ridge Model as reference since the MAE was slightly better at 0.14. Per our results we can infer that having a higher education (bachelors, masters, PhD) has the most impact along with looking for a job online through a job board (indeed, glassdoor, linkedin, etc.) The factors that played the least were gender and age

Our clustering model, GM, was not able to differentiate the cluster well with the data that was given. It has a low sillhouette score of 0.35, meaning there is not a lot of seperation nor density between the 3 clusters. Thus it would not be best to use this model to infer the different type of jobs using someones age and annual salary.

1.3.6 Sources

- Kaggle Submission by Shannon: For the dataset and a reference on how to have a similar theme
- plotnine documentation: On how to manipulate and enhance ggplot