Machine Learning Final Project: Mental Health in Tech Company

Chuanlong Pan, Lina Yu Dated:11/21/201

Introduction

Mental health, as defined by the Public Health Agency of Canada, is an individual's capacity to feel, think, and act in ways to achieve a better quality of life whilst respecting the personal, social, and cultural boundaries. Mental health can affect daily living, relationships, and physical health. Mental health problems has unprecedented importance in today's world, especially in tech companies. Tech companies are developing at a fast speed and employees in these companies are often under intense pressure to contribute their skills and knowledge. Poor mental health can cause a range of physical illness and thus has a direct impact on companies' business and revenue. From a manager's point of view, it's important to know how to improve employees' mental health and thus run business better.

Open Sourcing Mental Illness is a non-profit corporation dedicated to raising awareness, educating and providing resources to support mental wellness in the tech and open source communities. Every year, OSMI conducts a survey to measure employees' attitudes towards mental health in the tech workplace and examines the frequency of mental health disorders among tech workers.

This project aims to explore wheather we can predict a personal's possibility to seek treatment for mental illness based on information from a survey conducted by OSMI in 2014.

In [199]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Prep
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import binarize, LabelEncoder

# Models
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score, mean_squared_error, precis
from sklearn.model_selection import cross_val_score
from imblearn.over_sampling import SMOTE
from sklearn.metrics import confusion_matrix, accuracy_score, classifi
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

In [200]:
```
df = pd.read_csv("survey.csv")
```

In [201]:
```
df
```

Out[201]:

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	worl
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	
...	...	...	...	...	...	...	...	...	
1254	2015-09-12 11:17:21	26	male	United Kingdom	NaN	No	No	Yes	
1255	2015-09-26 01:07:35	32	Male	United States	IL	No	Yes	Yes	
1256	2015-11-07 12:36:58	34	male	United States	CA	No	Yes	Yes	S
1257	2015-11-30 21:25:06	46	f	United States	NC	No	No	No	
1258	2016-02-01 23:04:31	25	Male	United States	IL	No	Yes	Yes	S

1259 rows × 27 columns

In [202]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Timestamp                  1259 non-null   object
 1   Age                        1259 non-null   int64 
 2   Gender                     1259 non-null   object
 3   Country                    1259 non-null   object
 4   state                      744 non-null    object
 5   self_employed              1241 non-null   object
 6   family_history             1259 non-null   object
 7   treatment                  1259 non-null   object
 8   work_interfere             995 non-null    object
 9   no_employees               1259 non-null   object
 10  remote_work                1259 non-null   object
 11  tech_company               1259 non-null   object
 12  benefits                   1259 non-null   object
 13  care_options               1259 non-null   object
 14  wellness_program           1259 non-null   object
 15  seek_help                  1259 non-null   object
 16  anonymity                  1259 non-null   object
 17  leave                      1259 non-null   object
 18  mental_health_consequence  1259 non-null   object
 19  phys_health_consequence    1259 non-null   object
 20  coworkers                  1259 non-null   object
 21  supervisor                 1259 non-null   object
 22  mental_health_interview    1259 non-null   object
 23  phys_health_interview      1259 non-null   object
 24  mental_vs_physical         1259 non-null   object
 25  obs_consequence            1259 non-null   object
 26  comments                   164 non-null    object
dtypes: int64(1), object(26)
memory usage: 265.7+ KB
```

This dataset has 27 columns. Information of this survey can be split into 4 categories:

1. **Geographic and Demographic information of responders**: e.g. Age, Gender, Country, family history of mental illness
2. **Relevant information about responder's workspace**: e.g. if the responder is self-employed or not, how many employees of the company, remote work or not
3. Workspaces' support for mental health problem:

- **benefits**: Does your employer provide mental health benefits?
- **care_options**: Do you know the options for mental health care your employer provides?
- **wellness_program**: Has your employer ever discussed mental health as part of an employee wellness program?
- **seek_help**: Does your employer provide resources to learn more about mental health issues and how to seek help?
- **anonymity**: Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources
- **leave**: How easy is it for you to take medical leave for a mental health condition?

4. The survey also asks some possible negative consuquence about mental health occur in workspaces:

- **mental_health_consequence**: Do you think that discussing a mental health issue with your employer would have negative consequences?
- **phys_health_consequence**: Do you think that discussing a physical health issue with your employer would have negative consequences?
- **coworkers**: Would you be willing to discuss a mental health issue with your coworkers?
- **supervisor**: Would you be willing to discuss a mental health issue with your direct supervisor(s)?
- **mental_health_interview**: Would you bring up a mental health issue with a potential employer in an interview?
- **phys_health_interview**: Would you bring up a physical health issue with a potential employer in an interview?
- **mental_vs_physical**: Do you feel that your employer takes mental health as seriously as physical health?
- obs_consequence: Have you heard of or observed negative consequences for coworkers with mental health conditions in your workplace?

Data Cleaning

Check Missing Value

In [203]:
```
raw_data = df.copy()
null_count = df.isnull().sum().sort_values(ascending=False)
null_percent = null_count/len(df.index)*100
null_summary = pd.concat([null_count, null_percent], axis = 1, keys =
print("Missing value count and percentage: ")
print(null_summary)
```

```
Missing value count and percentage:
                           Count  Percentage
comments                    1095   86.973789
state                        515   40.905481
work_interfere               264   20.969023
self_employed                 18    1.429706
seek_help                      0    0.000000
obs_consequence                0    0.000000
mental_vs_physical             0    0.000000
phys_health_interview          0    0.000000
mental_health_interview        0    0.000000
supervisor                     0    0.000000
coworkers                      0    0.000000
phys_health_consequence        0    0.000000
mental_health_consequence      0    0.000000
leave                          0    0.000000
anonymity                      0    0.000000
Timestamp                      0    0.000000
wellness_program               0    0.000000
Age                            0    0.000000
benefits                       0    0.000000
tech_company                   0    0.000000
remote_work                    0    0.000000
no_employees                   0    0.000000
treatment                      0    0.000000
family_history                 0    0.000000
Country                        0    0.000000
Gender                         0    0.000000
care_options                   0    0.000000
```

Drop Columns

We simply drop Timestamp column since time and date are irrelevant to the result of this project.

Because not all the responders wrote down their comments and about 87% comments were missing, we will drop the comments column.

In [204]:
```
df = df.drop("comments",axis = 1)
```

We can see the state column also contains many missing values. Let's look at the state column along with the Country column.

In [205]:
```
country_count = df.Country.value_counts()
country_percent = country_count/len(df.index)*100
country_summary = pd.concat([country_count, country_percent], axis = 1
print("Country value count and percentage: ")
print(country_summary)
```

```
Country value count and percentage:
                         Count  Percentage
United States              751   59.650516
United Kingdom             185   14.694202
Canada                      72    5.718824
Germany                     45    3.574265
Ireland                     27    2.144559
Netherlands                 27    2.144559
Australia                   21    1.667990
France                      13    1.032566
India                       10    0.794281
New Zealand                  8    0.635425
Poland                       7    0.555997
Switzerland                  7    0.555997
Sweden                       7    0.555997
Italy                        7    0.555997
South Africa                 6    0.476569
Belgium                      6    0.476569
Brazil                       6    0.476569
Israel                       5    0.397141
Singapore                    4    0.317712
Bulgaria                     4    0.317712
Austria                      3    0.238284
Finland                      3    0.238284
Mexico                       3    0.238284
Russia                       3    0.238284
Denmark                      2    0.158856
Greece                       2    0.158856
Colombia                     2    0.158856
Croatia                      2    0.158856
Portugal                     2    0.158856
Moldova                      1    0.079428
```

```
Georgia                        1    0.079428
Bahamas, The                   1    0.079428
China                          1    0.079428
Thailand                       1    0.079428
Czech Republic                 1    0.079428
Norway                         1    0.079428
Romania                        1    0.079428
Nigeria                        1    0.079428
Japan                          1    0.079428
Hungary                        1    0.079428
Bosnia and Herzegovina         1    0.079428
Uruguay                        1    0.079428
Spain                          1    0.079428
Zimbabwe                       1    0.079428
Latvia                         1    0.079428
Costa Rica                     1    0.079428
Slovenia                       1    0.079428
Philippines                    1    0.079428
```

Explore state column

In [206]:
```
df.state.unique()
```

Out[206]:
```
array(['IL', 'IN', nan, 'TX', 'TN', 'MI', 'OH', 'CA', 'CT', 'MD', 'NY
',
       'NC', 'MA', 'IA', 'PA', 'WA', 'WI', 'UT', 'NM', 'OR', 'FL', 'M
N',
       'MO', 'AZ', 'CO', 'GA', 'DC', 'NE', 'WV', 'OK', 'KS', 'VA', 'N
H',
       'KY', 'AL', 'NV', 'NJ', 'SC', 'VT', 'SD', 'ID', 'MS', 'RI', 'W
Y',
       'LA', 'ME'], dtype=object)
```

About 60% of responders come from US and responders from other contries only has a small portion. Responders will only fillin state information if they live in US, thus many state data are missing, so we will simply drop country and state columns.

In [207]:
```
df.drop(["Country", "state"],axis = 1, inplace = True)
```

Work Interfere

In [208]:
```
df['work_interfere'].value_counts()
```

Out[208]:
```
Sometimes    465
Never        213
Rarely       173
Often        144
Name: work_interfere, dtype: int64
```

work_interfere column answers the question "If you have a mental health condition, do you feel that it interferes with your work?" and has about 21% data missing. Some responders who are not sure about the answers will leave this question blank, so we just fill the missing value as "Don't know".

In [209]:
```
df['work_interfere'].fillna('Don\'t know', inplace = True)
df['work_interfere'].value_counts()
```

Out[209]:
```
Sometimes     465
Don't know    264
Never         213
Rarely        173
Often         144
Name: work_interfere, dtype: int64
```

Self-employed

In [210]:
```
df['self_employed'].value_counts()
```

Out[210]:
```
No     1095
Yes     146
Name: self_employed, dtype: int64
```

self_employed column has only 18 missing data and "No" constitutes majority of the answers. We will replace these missing value to 'No' and assume that they are not self-employed.

In [211]:
```
df['self_employed'].fillna("No", inplace = True)
df['self_employed'].value_counts()
```

Out[211]:
```
No     1113
Yes     146
Name: self_employed, dtype: int64
```

Gender

In [212]:
```
df['Gender'].value_counts()
```

Out[212]:

Male	615
male	206
Female	121
M	116
female	62
F	38
m	34
f	15
Make	4
Male	3
Woman	3
Cis Male	2
Man	2
Female (trans)	2
Female	2
Trans woman	1
msle	1
male leaning androgynous	1
Neuter	1
cis male	1
queer	1
Female (cis)	1
Mail	1
cis-female/femme	1
A little about you	1
Malr	1
p	1
femail	1
Cis Man	1
Guy (-ish) ^_^	1
Enby	1
Agender	1
Androgyne	1
Male-ish	1
maile	1
Trans-female	1
Cis Female	1
something kinda male?	1
Mal	1
Male (CIS)	1
queer/she/they	1
non-binary	1
Femake	1
woman	1
Nah	1
All	1
fluid	1
Genderqueer	1

```
ostensibly male, unsure what that really means      1
Name: Gender, dtype: int64
```

As you can see, there are lots of distinct response in Gender's input data, so we need to transform and rename any inputs with the same meaning.

In [213]:
```
gender = df['Gender'].str.lower()

gender = df['Gender'].unique()
male_str = ["male", "m", "male-ish", "maile", "mal", "male (cis)", "ma
trans_str = ["trans-female", "something kinda male?", "queer/she/they"
female_str = ["cis female", "f", "female", "woman",  "femake", "female

for (row, col) in df.iterrows():

    if str.lower(col.Gender) in male_str:
        df['Gender'].replace(to_replace=col.Gender, value='male', inpl

    if str.lower(col.Gender) in female_str:
        df['Gender'].replace(to_replace=col.Gender, value='female', in

    if str.lower(col.Gender) in trans_str:
        df['Gender'].replace(to_replace=col.Gender, value='trans', inp

stk_list = ['A little about you', 'p']
df = df[~df['Gender'].isin(stk_list)]

print(df['Gender'].unique())
```

```
['female' 'male' 'trans']
```

Age

There are some unreasonable age values in this column, such as age 5 which is under the minimum federal working age or extrmely large value over 100. We set an age range from 16 to 65 years old and drop those unreasonable values.

In [214]:
```
df['Age'].unique()
```

Out[214]:
```
array([         37,          44,          32,          31,           3
3,
                35,          39,          42,          23,           2
9,
                36,          27,          46,          41,           3
4,
                30,          40,          38,          50,           2
4,
                18,          28,          26,          22,           1
9,
                25,          45,          21,         -29,           4
3,
                56,          60,          54,         329,           5
5,
       99999999999,          48,          20,          57,           5
8,
                47,          62,          51,          65,           4
9,
             -1726,           5,          53,          61,           1
1,
                72])
```

In [215]:
```
df.drop(df[df['Age'] < 16].index, inplace=True)
df.drop(df[df['Age'] > 65].index, inplace=True)
print(np.sort(df['Age'].unique()))
```

```
[18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
41
 42 43 44 45 46 47 48 49 50 51 53 54 55 56 57 58 60 61 62 65]
```

Let's check again the missing value.

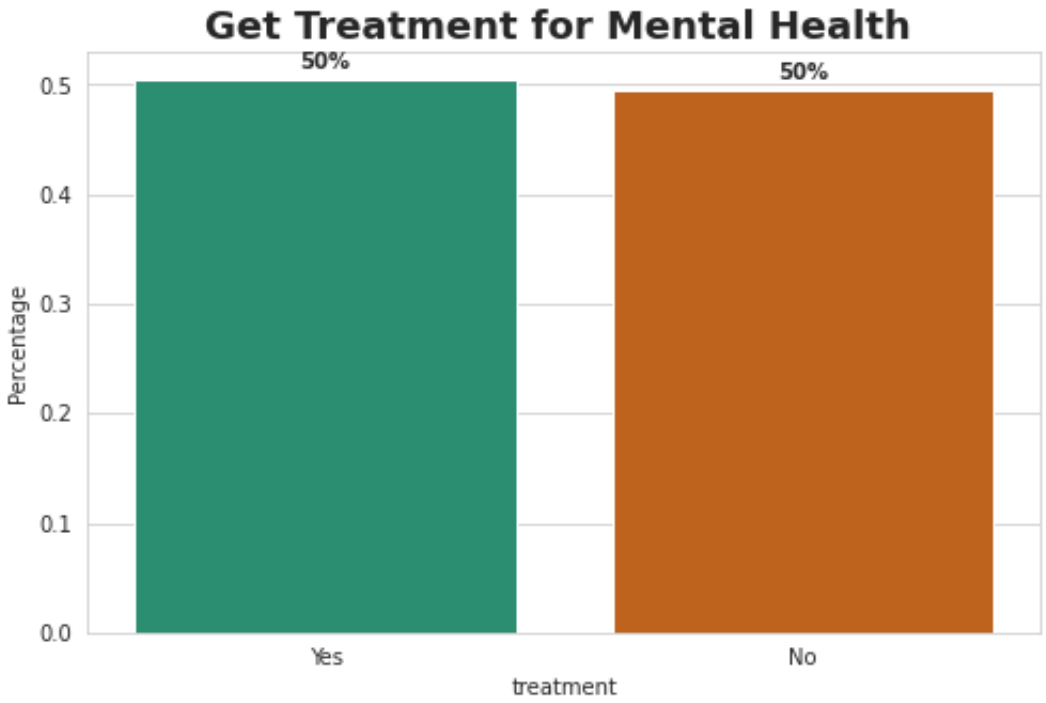
In [216]:
```
null_count = df.isnull().sum().sort_values(ascending=False)
null_percent = null_count/len(df.index)*100
null_summary = pd.concat([null_count, null_percent], axis = 1, keys =
print("Missing value count and percentage: ")
print(null_summary)
```

```
Missing value count and percentage:
                           Count  Percentage
Timestamp                      0         0.0
Age                            0         0.0
mental_vs_physical             0         0.0
phys_health_interview          0         0.0
mental_health_interview        0         0.0
supervisor                     0         0.0
coworkers                      0         0.0
phys_health_consequence        0         0.0
mental_health_consequence      0         0.0
leave                          0         0.0
anonymity                      0         0.0
seek_help                      0         0.0
wellness_program               0         0.0
care_options                   0         0.0
benefits                       0         0.0
tech_company                   0         0.0
remote_work                    0         0.0
no_employees                   0         0.0
work_interfere                 0         0.0
treatment                      0         0.0
family_history                 0         0.0
self_employed                  0         0.0
Gender                         0         0.0
obs_consequence                0         0.0
```

Exploratory Data Analysis

Target Variable

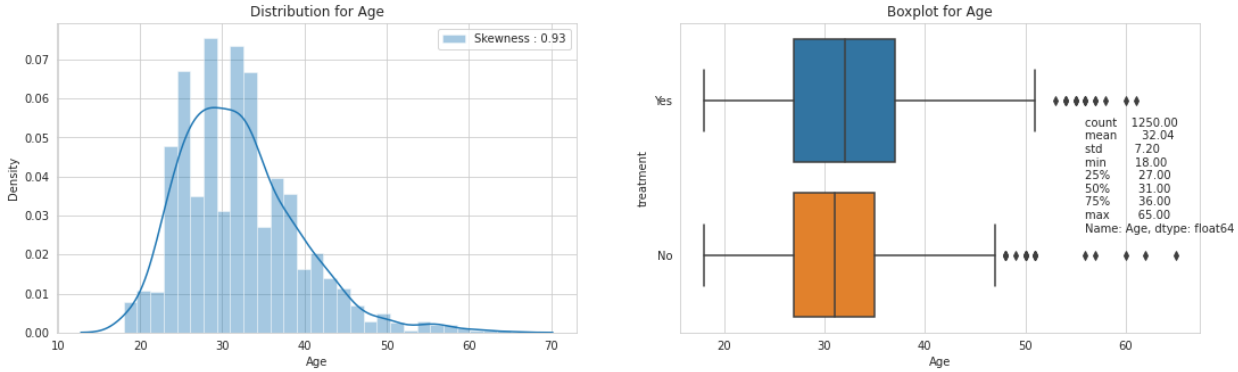
In [217]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (8,5))
plt.title('Get Treatment for Mental Health ', fontsize=18, fontweight=
eda_percentage = df['treatment'].value_counts(normalize = True).rename

ax = sns.barplot(x = 'treatment', y = 'Percentage', data = eda_percent
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.0%}', (x + width/2, y + height*1.02), ha='c
```



Our target variable is "treatment" which answers the question "Have you get treatment for a mental health condition?". From the diagram, the percentage of responders who have got treatment is 50% which is pretty high. It indicates that tech companies need to put more attention on promoting mental health and supporting employees on mental health issues.

Next, we'll explore the attributes from the four categories to see the relationship with the mental illness treatment.

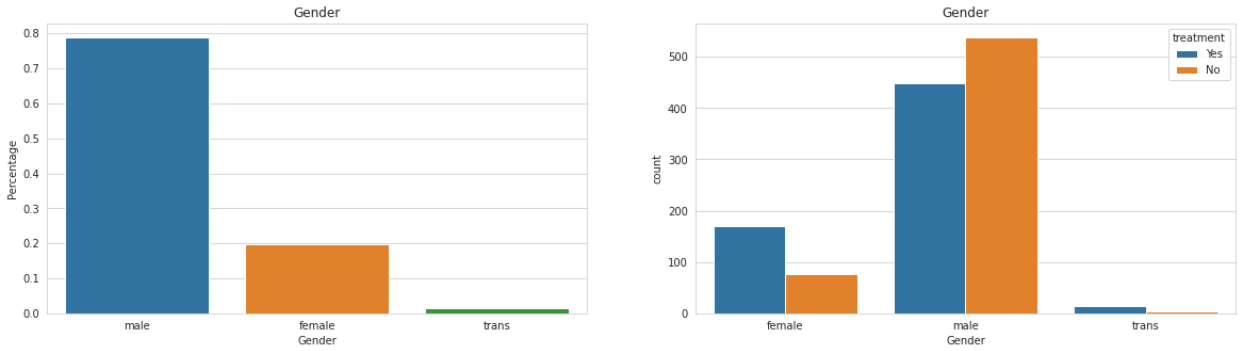
In [218]:
```
import warnings
warnings.filterwarnings("ignore")
plt.figure(figsize = (18,5))
plt.subplot(1,2,1)
sns.distplot(df['Age'], label = 'Skewness : %.2f'%(df['Age'].skew()))
plt.legend(loc = 0, fontsize = 10)
plt.title('Distribution for Age')
plt.subplot(1,2,2)
sns.boxplot(x = "Age", y = "treatment", data = df)
plt.title('Boxplot for Age')
age = str(df['Age'].describe().round(2))
plt.text(56, 0.85, age)
plt.show()
```



In this survey, most employees age 25 to early 40s. The distribution of ages is right-skewed which matches the tech industry tends to have younger employees.

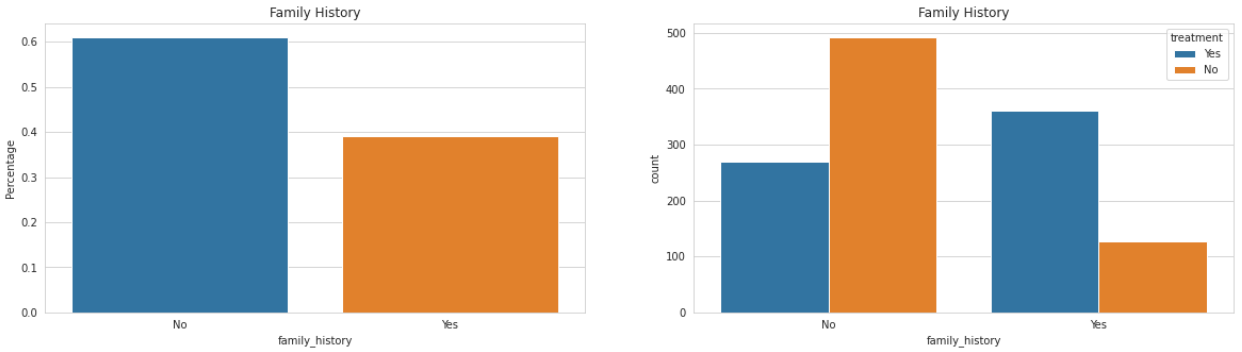
From the boxplot above, there is no huge difference of age between employees who get treattment or not.

In [219]:
```
warnings.filterwarnings("ignore")
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
percentage = df['Gender'].value_counts(normalize = True).rename_axis('
sns.barplot(x = 'Gender', y = 'Percentage', data = percentage.head(10)
plt.title('Gender')
plt.subplot(1,2,2)
sns.countplot(df['Gender'], hue = df['treatment'])
plt.title('Gender')
plt.show()
```



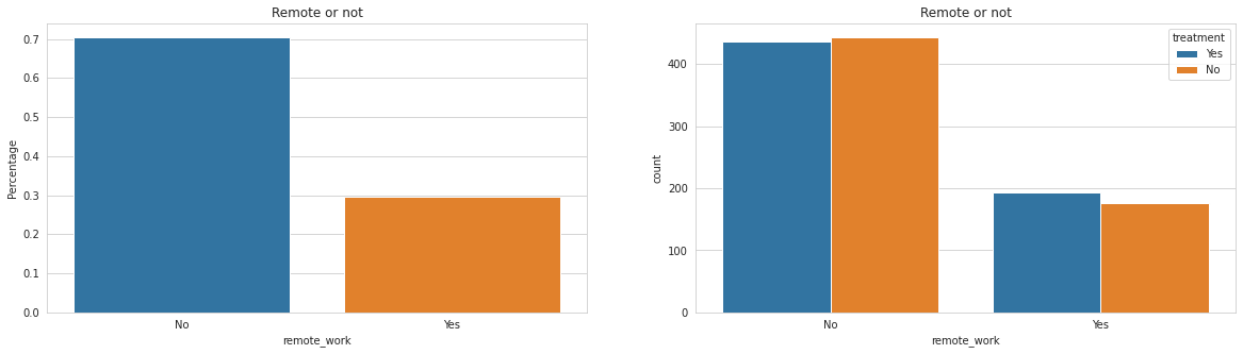
Almost 80% of responders are male. However, larger proportion of female responders have sought treatments than male responders. That could be caused by the pressure from large gap between the proportion of male and female employees in the tech workspaces.

In [220]:
```
warnings.filterwarnings("ignore")
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
percentage = df['family_history'].value_counts(normalize = True).renam
sns.barplot(x = 'family_history', y = 'Percentage', data = percentage)
plt.title('Family History')
plt.subplot(1,2,2)
sns.countplot(df['family_history'], hue = df['treatment'])
plt.title('Family History')
plt.show()
```



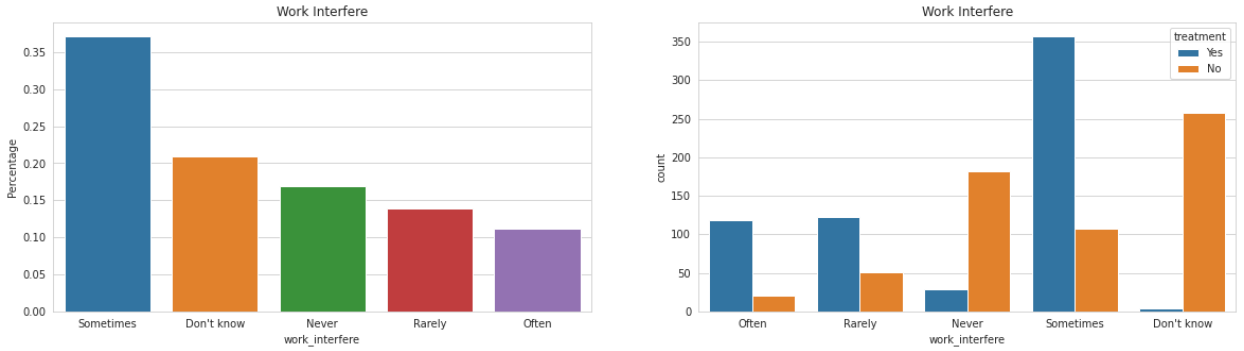
Around 40% of responders have a family history of mental problem. These responders with family history of mental illness were more likely to seek treatment than the other groups, because family history is a significant factor for mental health.

In [221]:
```
warnings.filterwarnings("ignore")
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
percentage = df['remote_work'].value_counts(normalize = True).rename_a
sns.barplot(x = 'remote_work', y = 'Percentage', data = percentage)
plt.title('Remote or not')
plt.subplot(1,2,2)
sns.countplot(df['remote_work'], hue = df['treatment'])
plt.title('Remote or not')
plt.show()
```



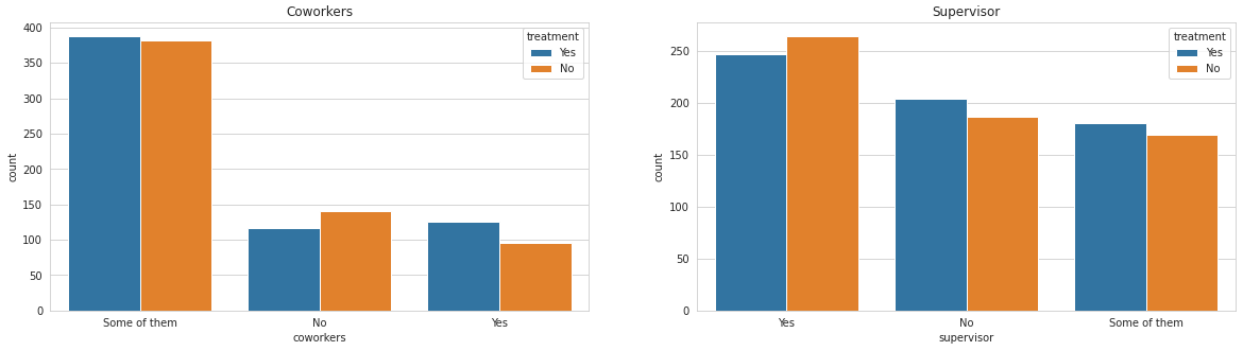
The survey was conducted in 2014. Only about 30% responders worked remotely at that time. For those who worked remotely, the percentage of seeking treatment is slightly higher.

In [222]:
```
warnings.filterwarnings("ignore")
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
percentage = df['work_interfere'].value_counts(normalize = True).renam
sns.barplot(x = 'work_interfere', y = 'Percentage', data = percentage)
plt.title('Work Interfere')
plt.subplot(1,2,2)
sns.countplot(df['work_interfere'], hue = df['treatment'])
plt.title('Work Interfere')
plt.show()
```



For work interfere, around 37% say if they have mental health condition, they feel that it interferes with their work sometimes. Larger porportion of this group would seek treatment.

In [223]:
```
warnings.filterwarnings("ignore")
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
sns.countplot(df['coworkers'], hue = df['treatment'])
plt.title('Coworkers')
plt.subplot(1,2,2)
sns.countplot(df['supervisor'], hue = df['treatment'])
plt.title('Supervisor')
plt.show()
```



Regarding the question, 'Would you be willing to discuss a mental health issue with your coworkers?', responders who answer yes have higher possibility to get treatment.

The survey also asks if the responders are willing to discuss a mental health issue with their direct supervisor.There are more people say yes to discuss with their supervisors, over 50% of them do not get a treatment. Talking to someone with a higher level may be helpful for their mental health condiction.

Encoding

In [224]:
```
list_col=['Age', 'Gender', 'self_employed', 'family_history', 'treatme
       'work_interfere', 'no_employees', 'remote_work', 'tech_company'
       'benefits', 'care_options', 'wellness_program', 'seek_help',
       'anonymity', 'leave', 'mental_health_consequence',
       'phys_health_consequence', 'coworkers', 'supervisor',
       'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'obs_consequence']

for col in list_col:
    print('{} :{} ' . format(col.upper(),df[col].unique()))
```

```
AGE :[37 44 32 31 33 35 39 42 23 29 36 27 46 41 34 30 40 38 50 24 18
28 26 22
 19 25 45 21 43 56 60 54 55 48 20 57 58 47 62 51 65 49 53 61]
GENDER :['female' 'male' 'trans']
SELF_EMPLOYED :['No' 'Yes']
FAMILY_HISTORY :['No' 'Yes']
TREATMENT :['Yes' 'No']
WORK_INTERFERE :['Often' 'Rarely' 'Never' 'Sometimes' "Don't know"]
NO_EMPLOYEES :['6-25' 'More than 1000' '26-100' '100-500' '1-5' '500-
1000']
REMOTE_WORK :['No' 'Yes']
TECH_COMPANY :['Yes' 'No']
BENEFITS :['Yes' "Don't know" 'No']
CARE_OPTIONS :['Not sure' 'No' 'Yes']
WELLNESS_PROGRAM :['No' "Don't know" 'Yes']
SEEK_HELP :['Yes' "Don't know" 'No']
ANONYMITY :['Yes' "Don't know" 'No']
LEAVE :['Somewhat easy' "Don't know" 'Somewhat difficult' 'Very diffi
cult'
 'Very easy']
MENTAL_HEALTH_CONSEQUENCE :['No' 'Maybe' 'Yes']
PHYS_HEALTH_CONSEQUENCE :['No' 'Yes' 'Maybe']
COWORKERS :['Some of them' 'No' 'Yes']
SUPERVISOR :['Yes' 'No' 'Some of them']
MENTAL_HEALTH_INTERVIEW :['No' 'Yes' 'Maybe']
PHYS_HEALTH_INTERVIEW :['Maybe' 'No' 'Yes']
MENTAL_VS_PHYSICAL :['Yes' "Don't know" 'No']
OBS_CONSEQUENCE :['No' 'Yes']
```

In [225]:
```
le = LabelEncoder()
dfencode = df.copy()
```

In [226]:
```
object_cols = ['Gender', 'self_employed', 'family_history', 'treatment
       'work_interfere', 'no_employees', 'remote_work', 'tech_company'
       'benefits', 'care_options', 'wellness_program', 'seek_help',
       'anonymity', 'leave', 'mental_health_consequence',
       'phys_health_consequence', 'coworkers', 'supervisor',
       'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'obs_consequence']
for col in object_cols:
    le.fit(dfencode[col])
    dfencode[col] = le.transform(dfencode[col])
```

In [227]:
```
dfencode.head()
```

Out[227]:

	Timestamp	Age	Gender	self_employed	family_history	treatment	work_interfere	no_empl
0	2014-08-27 11:29:31	37	0	0	0	1	2	
1	2014-08-27 11:29:37	44	1	0	0	0	3	
2	2014-08-27 11:29:44	32	1	0	0	0	3	
3	2014-08-27 11:29:46	31	1	0	1	1	2	
4	2014-08-27 11:30:22	31	1	0	0	0	1	

5 rows × 24 columns

Run correlation heatmap

It is important to see whether the dependent variables have any relationships with target variable.

In [228]:
```
corr = dfencode.corr(method="spearman")
corr
```

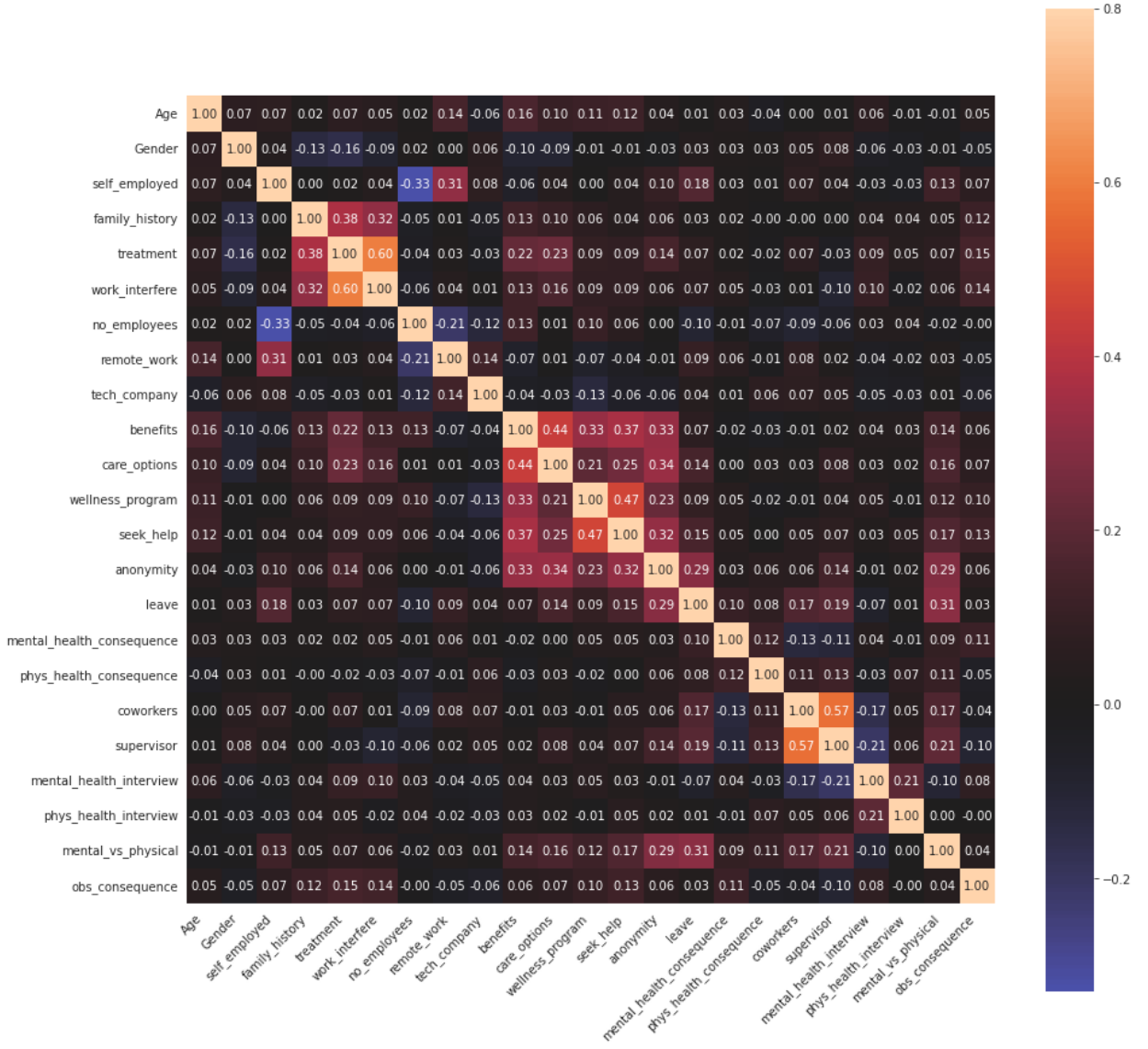
Out[228]:

	Age	Gender	self_employed	family_history	treatment	wor
Age	1.000000	0.067082	0.074379	0.020262	0.068120	
Gender	0.067082	1.000000	0.035909	-0.131841	-0.157417	
self_employed	0.074379	0.035909	1.000000	0.002911	0.016733	
family_history	0.020262	-0.131841	0.002911	1.000000	0.376067	
treatment	0.068120	-0.157417	0.016733	0.376067	1.000000	
work_interfere	0.050687	-0.088686	0.040300	0.316552	0.598370	
no_employees	0.016107	0.024291	-0.330170	-0.046526	-0.042632	
remote_work	0.142302	0.003100	0.314619	0.012760	0.025321	
tech_company	-0.057162	0.061665	0.075858	-0.047636	-0.030903	
benefits	0.157818	-0.097850	-0.061450	0.128271	0.224420	
care_options	0.103276	-0.094725	0.036717	0.103849	0.232769	
wellness_program	0.114399	-0.008125	0.001164	0.062801	0.093025	
seek_help	0.120450	-0.005046	0.042656	0.040409	0.086911	
anonymity	0.039896	-0.026185	0.102070	0.063607	0.144256	
leave	0.007599	0.034225	0.178654	0.026732	0.071898	
mental_health_consequence	0.032821	0.029412	0.025741	0.019634	0.019651	
phys_health_consequence	-0.036222	0.026126	0.010170	-0.004990	-0.015956	
coworkers	0.003710	0.053775	0.073500	-0.002673	0.070142	
supervisor	0.007219	0.076461	0.036865	0.001512	-0.034721	
mental_health_interview	0.062741	-0.055973	-0.028039	0.041051	0.094681	
phys_health_interview	-0.009706	-0.031482	-0.030967	0.041386	0.049311	
mental_vs_physical	-0.012938	-0.011733	0.125921	0.046671	0.070932	
obs_consequence	0.046345	-0.047310	0.068590	0.115500	0.151016	

23 rows × 23 columns

In [229]:
```
f, ax = plt.subplots(figsize=(15, 15))
sns.heatmap(corr,
            vmax=.8,
            center = 0,
            square=True
```

```
            square=True,
            cbar=True,
            annot=True,
            fmt='.2f',
            annot_kws={'size': 10},
            color='green'
           )

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
plt.show()
```



From the graph above, there is almost no relationship between our target variable "treatment" and other variables like "Age", "self_employeed", "remote_work", "tech_company", "no_employees", "physical_health_consequence", "coworkers", "phys_health_interview", "mental_vs_physical". Also, some variables such as "wellness_program", "seek_help", "anonymity", "obs_consequence", "mental_health_interview" have weak relationship with "tratment" about 10%. "family_history", "benefits", "care_options" have rilatively stronger relationship around 20% to 40%. "Gender" has a weak negatively relationship with the target variable. Lastly, there is a strong relationship between treatment and work_interfere with 63%.

In [230]:
```
corr_mat = dfencode.corr()
corr_mat
```

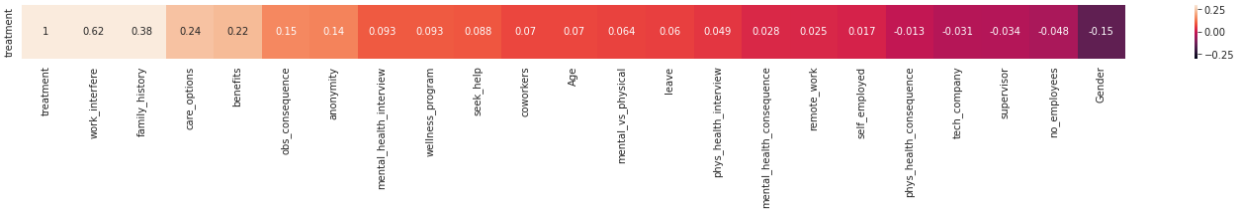
Out[230]:

	Age	Gender	self_employed	family_history	treatment	wor
Age	1.000000	0.067751	0.075854	0.003675	0.070164	
Gender	0.067751	1.000000	0.035267	-0.128249	-0.151373	
self_employed	0.075854	0.035267	1.000000	0.002911	0.016733	
family_history	0.003675	-0.128249	0.002911	1.000000	0.376067	
treatment	0.070164	-0.151373	0.016733	0.376067	1.000000	
work_interfere	0.040788	-0.089456	0.038597	0.323940	0.616119	
no_employees	0.029288	0.026645	-0.335913	-0.050156	-0.048147	
remote_work	0.142003	0.002027	0.314619	0.012760	0.025321	
tech_company	-0.049431	0.059041	0.075858	-0.047636	-0.030903	
benefits	0.152279	-0.093233	-0.055376	0.126620	0.224037	
care_options	0.109536	-0.089779	0.037356	0.105083	0.235477	
wellness_program	0.109810	-0.005029	0.001504	0.062849	0.092857	
seek_help	0.127741	-0.006418	0.037509	0.041909	0.088047	
anonymity	0.027887	-0.022193	0.097806	0.060630	0.143013	
leave	-0.011632	0.035631	0.168266	0.017920	0.060104	
mental_health_consequence	0.032221	0.026937	0.020526	0.025673	0.027956	
phys_health_consequence	-0.045506	0.019443	0.012023	-0.001538	-0.013122	
coworkers	-0.007608	0.053414	0.073675	-0.002505	0.070239	
supervisor	0.004687	0.072111	0.036628	0.001481	-0.034300	
mental_health_interview	0.062833	-0.045849	-0.024563	0.037824	0.093321	
phys_health_interview	-0.024435	-0.015080	-0.026736	0.035344	0.049111	
mental_vs_physical	-0.007535	-0.010654	0.131839	0.040050	0.063594	
obs_consequence	0.061374	-0.044123	0.068590	0.115500	0.151016	

23 rows × 23 columns

In [231]:
```
corr_treatment = corr_mat['treatment'].sort_values(ascending=False).to
plt.subplots(figsize=(25,1))
sns.heatmap(corr_treatment, vmin = -.3, vmax=.3, annot=True)
```

Out[231]: <AxesSubplot:>



The top 9 highly correlated with treatment is: work_interfere, family history, care options, Gender, obs consequence, benefits, leave, mental health consequence, anonymity

For Gender variable, we can see female gender is positively correlated with treatment, while male gender is the opposite: negatively correlated the "treatment" target. That is consistent with our observation in preliminary analysis that female responders tend to seek treatment more often.

The top 9 variables are also consistent with our preliminary analysis.

Check the correlations between the 9 variables.

In [232]:
```
Top_corr=['work_interfere', 'family_history', 'care_options', 'benefit
       'obs_consequence', 'anonymity', 'mental_health_interview',
       'wellness_program', 'Gender']
```

Model

Prediction Model

I will try these base models and ensemble models to predict whether or not a responder has sought for treatment in a mental health condition from these features. For this task, I evaluated the following algorithms:

Random Forest: This model uses lots of decision trees to classifies the dataset into smaller subsets, and to define a conclusion about a target value. The tree consists of leaves, where the intermediate ones are the decision nodes and the ones from the extremes are the final outcomes. This model was chosen, because it can be easily interpreted, visualized and explained. Also due the fact that this model implicitly perform variable screening or feature selection.

Naive Bayes： (GaussianNB): This model is a classification technique based on the Bayes' Theorem. It assumes the independence among the involved features. Nevertheless, this approach performers well even on data that are dependent between them. This algorithm was created by Bayes to prove the existence of God. It relies on the probability of an event, based on prior knowledge of conditions that might be related to the event. This model was chosen, because this family of algorithms can predict well with small set of data and when there is a large number of features comparatively.

K-nearest neighbors (KNN): This algorithm takes in consideration the k closest points (neighbors) around the target and use them learn how to classify the desired point. This model was chosen, because its simple to implement, no assumption about the data is necessary and the non-parametric nature of KNN gives an advantage in certain settings where the data may be highly unusual.

XGBoost: This is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

LDA: is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of distinct categories. LDA is most commonly used to discover a user-specified number of topics shared by documents within a text corpus.This model was chosen, because provides probabilities for outcomes and a convenient probability scores for observations.

Neural Network: adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

Model

Random Forest, Gaussian naive Bayes, K-nearest neighbors, Logistic regression, XGBoost, LDA, Neural Network

In [233]:
```
X=dfencode[['work_interfere', 'family_history', 'care_options', 'benef
       'obs_consequence', 'anonymity', 'mental_health_interview',
       'wellness_program', 'seek_help']]
```

In [234]:
```
X
```

Out[234]:

	work_interfere	family_history	care_options	benefits	obs_consequence	anonymity	men
0	2	0	1	2	0	2	
1	3	0	0	0	0	0	
2	3	0	0	1	0	0	
3	2	1	2	1	1	1	
4	1	0	0	2	0	0	
...	...	...	...	...	...	...	
1254	0	0	0	1	0	0	
1255	2	1	2	2	0	2	
1256	4	1	2	2	0	0	
1257	0	0	2	1	0	0	
1258	4	1	2	2	0	2	

1250 rows × 9 columns

In [235]:
```
y = dfencode["treatment"]
```

In [236]:
```
y
```

Out[236]:
```
0       1
1       0
2       0
3       1
4       0
       ..
1254    1
1255    1
1256    1
1257    0
1258    1
Name: treatment, Length: 1250, dtype: int64
```

Feature Normalization

In [237]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Training / Testing set split

In [238]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =0
```

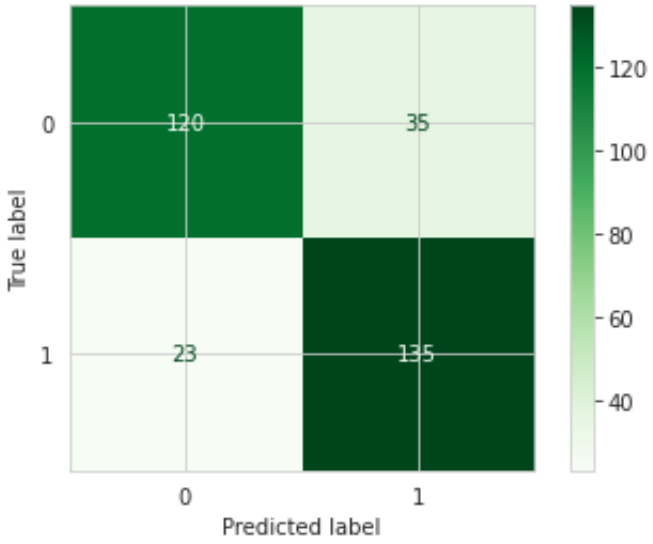
In [239]:
```
X_test
```

Out[239]:
```
array([[-0.18355047, -0.80026242,  1.21166515, ..., -2.03968502,
         1.68540583,  1.58104851],
       [-0.81043048, -0.80026242,  1.21166515, ...,  0.31235029,
        -1.79971789, -1.31040817],
       [ 1.07020956,  1.2495901 ,  0.05549611, ...,  0.31235029,
        -0.05715603,  0.13532017],
       ...,
       [ 1.07020956, -0.80026242, -1.10067292, ...,  0.31235029,
        -0.05715603,  0.13532017],
       [ 1.07020956, -0.80026242,  1.21166515, ...,  0.31235029,
         1.68540583,  1.58104851],
       [-0.81043048,  1.2495901 ,  1.21166515, ...,  0.31235029,
        -1.79971789,  1.58104851]])
```

Random Forest

In [240]:
```
RandomForest = RandomForestClassifier()
model_rf = RandomForest.fit(X_train, y_train)
pred = model_rf.predict(X_test)
print(classification_report(y_true = y_test, y_pred = pred))
plot_confusion_matrix(model_rf, X = X_test, y_true = y_test,cmap = 'Gr
```

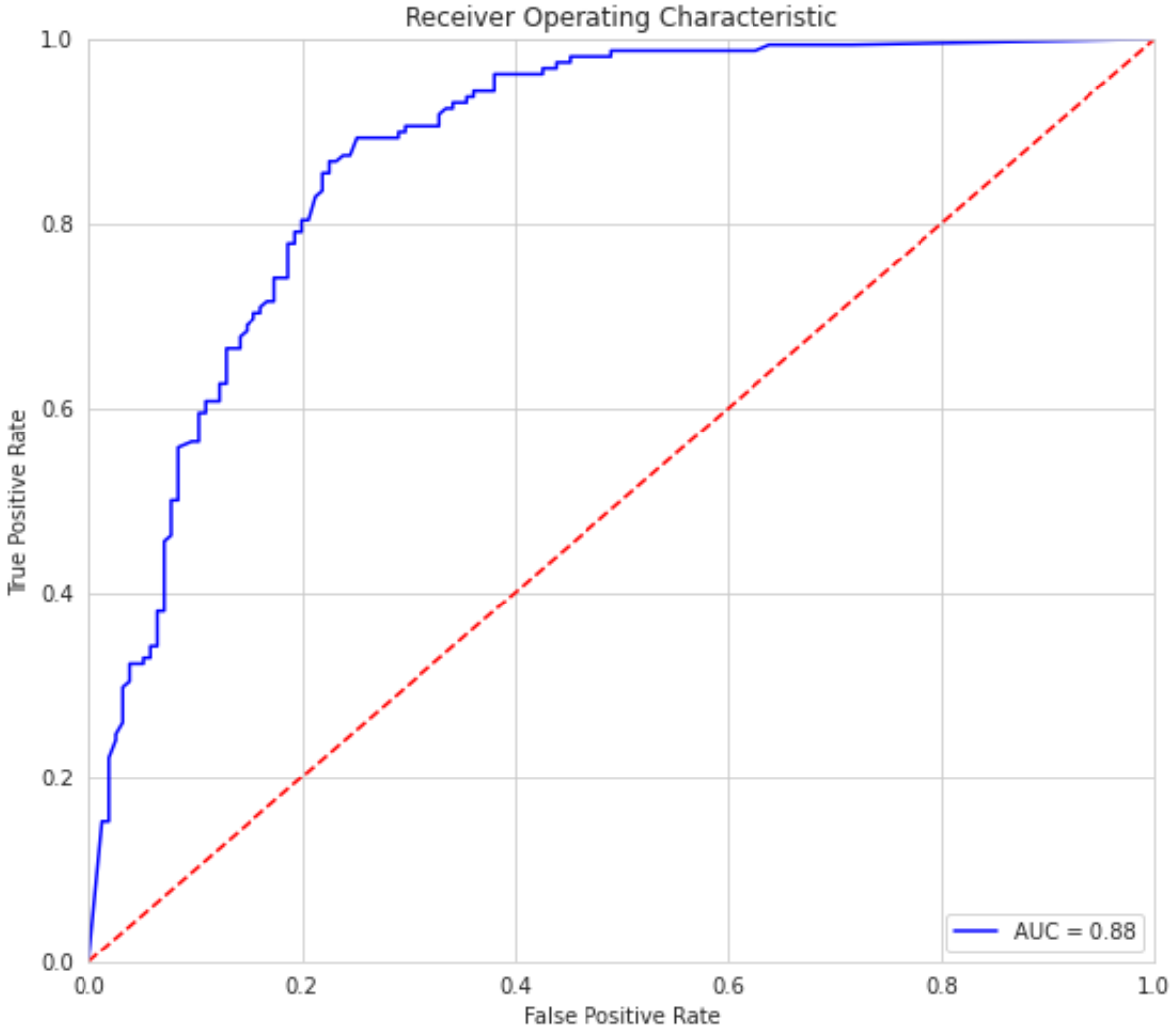
```
              precision    recall  f1-score   support

           0       0.84      0.77      0.81       155
           1       0.79      0.85      0.82       158

    accuracy                           0.81       313
   macro avg       0.82      0.81      0.81       313
weighted avg       0.82      0.81      0.81       313
```



In [241]:
```
accuracy_rf=accuracy_score(y_test,pred)
recall=recall_score(y_test,pred)
precision=precision_score(y_test,pred)
print("Random forest's accuracy = ",accuracy_rf )
print("Random forest's recall= ",recall)
print("Random forest's precision= ",precision,"\n" )
```

```
Random forest's accuracy =  0.8146964856230032
Random forest's recall=  0.8544303797468354
Random forest's precision=  0.7941176470588235
```

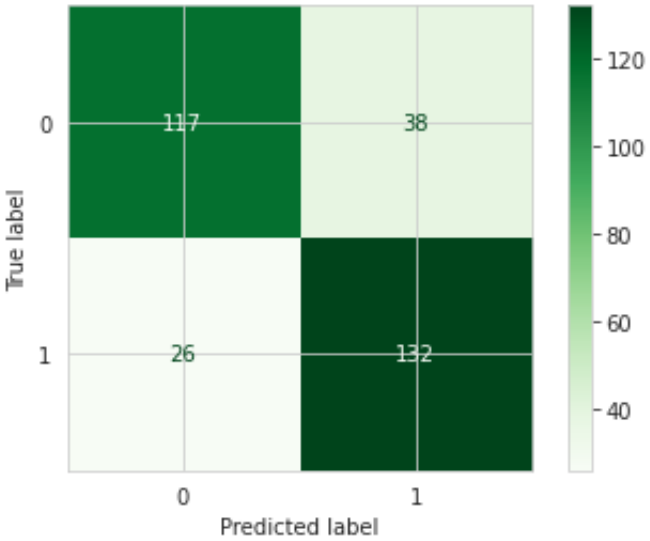
In [242]:
```
pred_prob = model_rf.predict_proba(X_test)
pred_prob = pred_prob[:,1]
metrics.roc_auc_score(y_test, pred_prob)
fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_prob)
roc_auc = metrics.auc(fpr, tpr)
plt.figure(figsize = (9,8))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



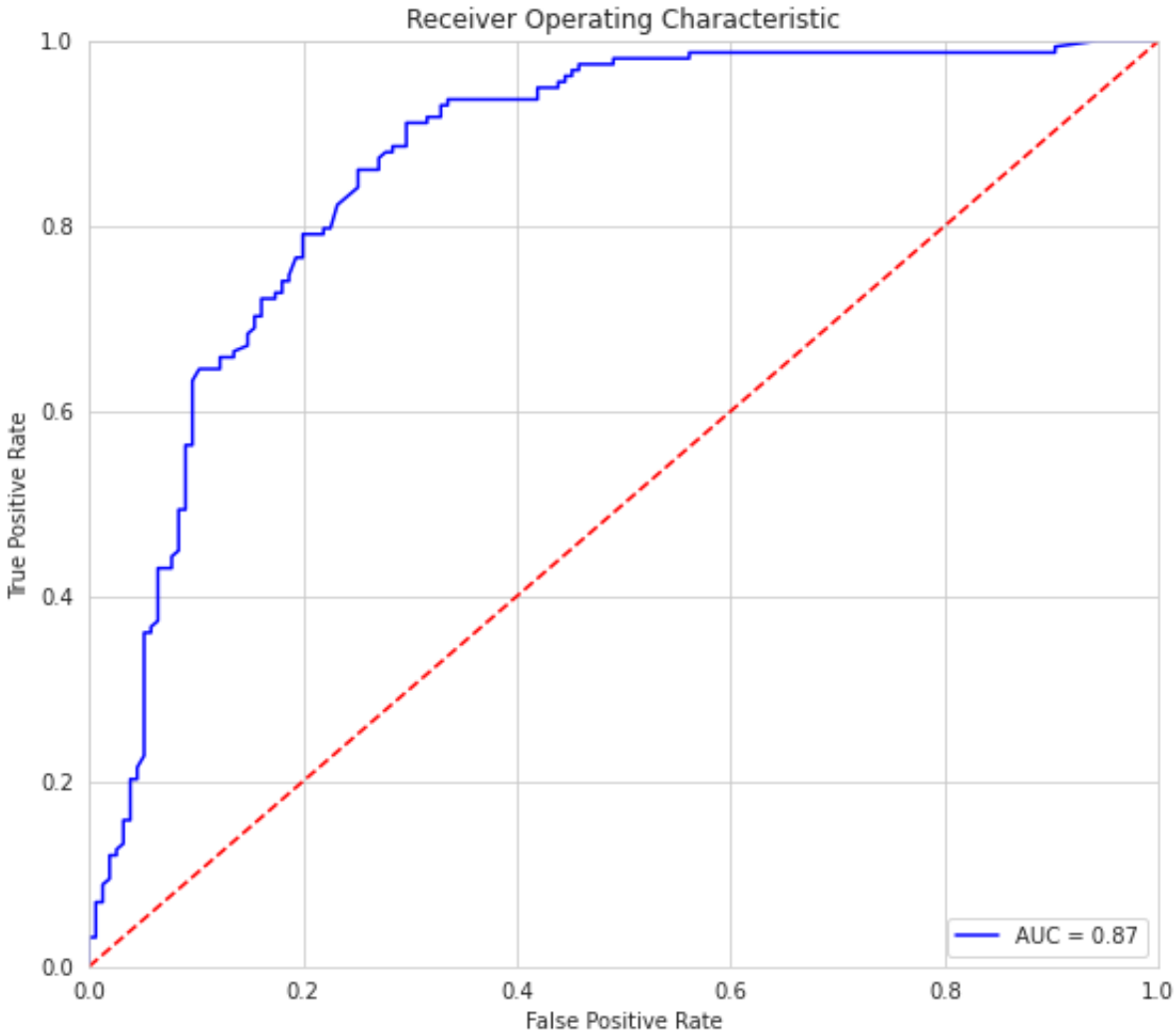
XGBoost - Classification

In [243]:
```
warnings.filterwarnings("ignore")
model_xgb = XGBClassifier()
model_xgb.fit(X_train, y_train)
pred = model_xgb.predict(X_test)
print(classification_report(y_true = y_test, y_pred = pred))
plot_confusion_matrix(model_xgb, X = X_test, y_true = y_test, cmap = '
accuracy_xgb=accuracy_score(y_test,pred)

recall=recall_score(y_test,pred)
precision=precision_score(y_test,pred)


print("XGBoost's accuracy = ",accuracy_xgb )
print("XGBoost's recall= ",recall)
print("XGBoost's precision= ",precision,"\n" )
```

```
[09:09:31] WARNING: ../src/learner.cc:1095: Starting in XGBoost 1.3.0
, the default evaluation metric used with the objective 'binary:logis
tic' was changed from 'error' to 'logloss'. Explicitly set eval_metri
c if you'd like to restore the old behavior.
              precision    recall  f1-score   support

           0       0.82      0.75      0.79       155
           1       0.78      0.84      0.80       158

    accuracy                           0.80       313
   macro avg       0.80      0.80      0.80       313
weighted avg       0.80      0.80      0.80       313

XGBoost's accuracy =  0.7955271565495208
XGBoost's recall=  0.8354430379746836
XGBoost's precision=  0.7764705882352941
```



In [244]:
```
pred_prob = model_xgb.predict_proba(X_test)
pred_prob = pred_prob[:,1]
metrics.roc_auc_score(y_test, pred_prob)
fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_prob)
roc_auc = metrics.auc(fpr, tpr)
plt.figure(figsize = (9,8))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

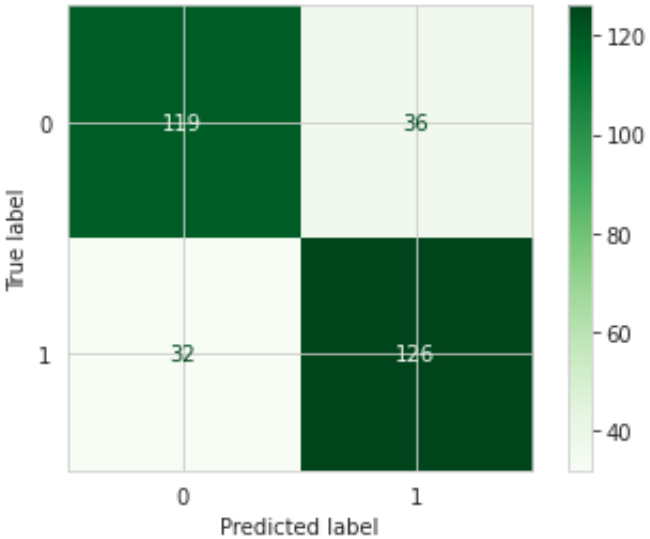


Naive Bayes

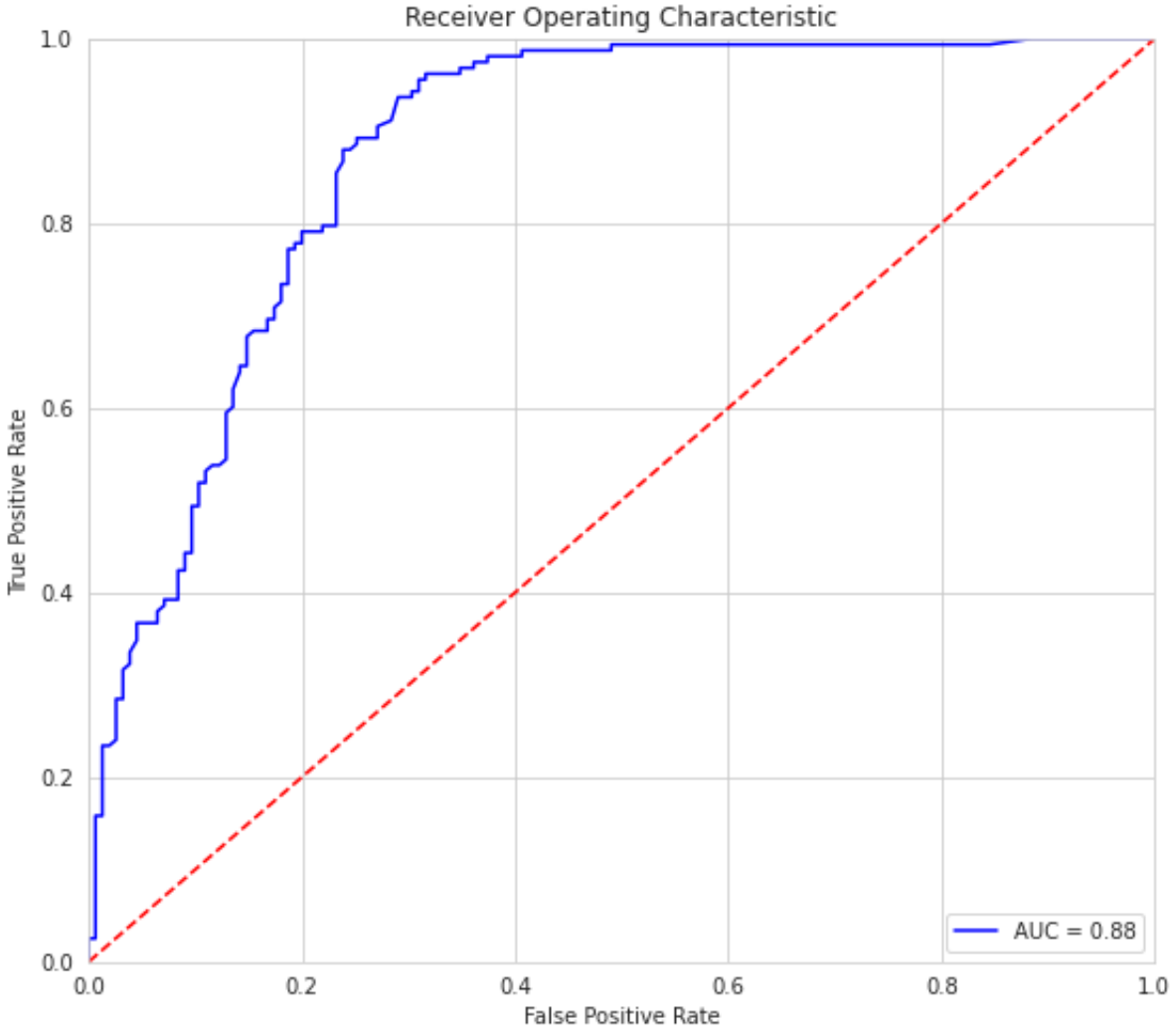
In [245]:
```
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)
pred = model_nb.predict(X_test)
```

In [246]:
```
pred  = model_nb.predict(X_test)
print(classification_report(y_true = y_test, y_pred = pred))
plot_confusion_matrix(model_nb, X = X_test, y_true = y_test, cmap='Gre
accuracy_nb=accuracy_score(y_test,pred)
recall=recall_score(y_test,pred)
precision=precision_score(y_test,pred)
print("Naive Bayes' accuracy = ",accuracy_nb )
print("Naive Bayes's recall= ",recall)
print("Naive Bayes's precision= ",precision,"\n" )
```

```
              precision    recall  f1-score   support

           0       0.79      0.77      0.78       155
           1       0.78      0.80      0.79       158

    accuracy                           0.78       313
   macro avg       0.78      0.78      0.78       313
weighted avg       0.78      0.78      0.78       313

Naive Bayes' accuracy =  0.7827476038338658
Naive Bayes's recall=  0.7974683544303798
Naive Bayes's precision=  0.7777777777777778
```



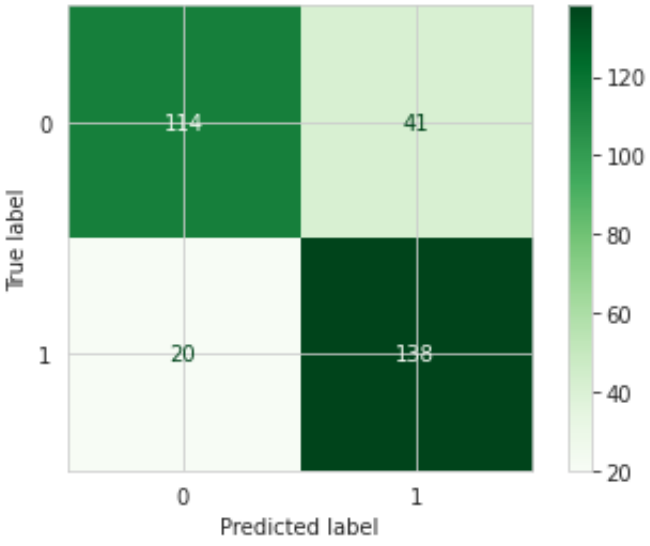
In [247]:
```
pred_prob = model_nb.predict_proba(X_test)
pred_prob = pred_prob[:,1]
metrics.roc_auc_score(y_test, pred_prob)
fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_prob)
roc_auc = metrics.auc(fpr, tpr)
plt.figure(figsize = (9,8))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



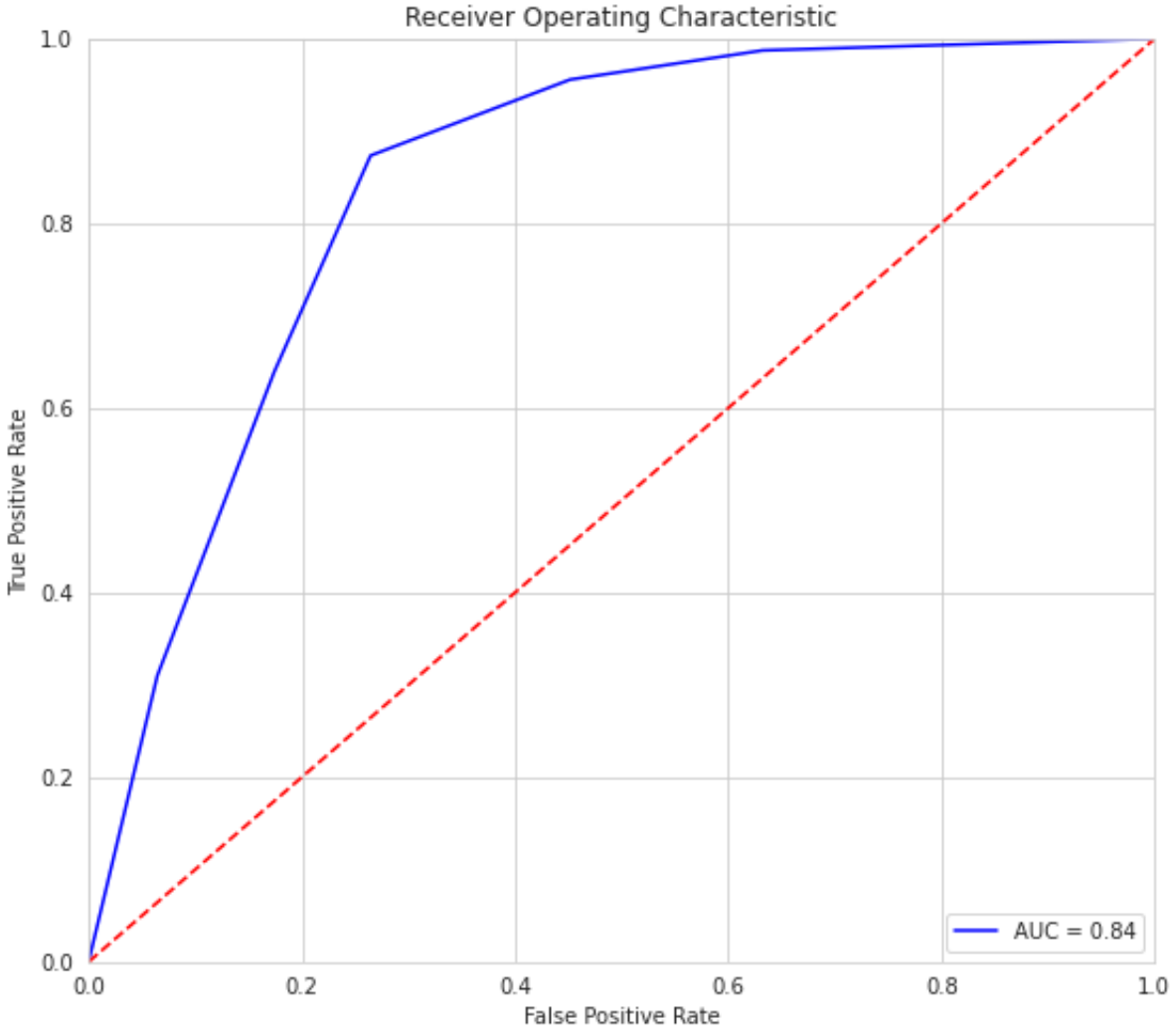
Knn

In [248]:
```
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=5)
model_knn.fit(X_train, y_train)
pred = model_knn.predict(X_test)
print(classification_report(y_true = y_test, y_pred = pred))
plot_confusion_matrix(model_knn, X = X_test, y_true = y_test, cmap='Gr
accuracy_nb=accuracy_score(y_test,pred)
recall=recall_score(y_test,pred)
precision=precision_score(y_test,pred)
print("KNN accuracy = ",accuracy_nb )
print("KNN's recall= ",recall)
print("KNN's precision= ",precision,"\n" )
```

```
              precision    recall  f1-score   support

           0       0.85      0.74      0.79       155
           1       0.77      0.87      0.82       158

    accuracy                           0.81       313
   macro avg       0.81      0.80      0.80       313
weighted avg       0.81      0.81      0.80       313

KNN accuracy =  0.805111821086262
KNN's recall=  0.8734177215189873
KNN's precision=  0.770949720670391
```



In [249]:
```
pred_prob = model_knn.predict_proba(X_test)
pred_prob = pred_prob[:,1]
metrics.roc_auc_score(y_test, pred_prob)
fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_prob)
roc_auc = metrics.auc(fpr, tpr)
plt.figure(figsize = (9,8))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

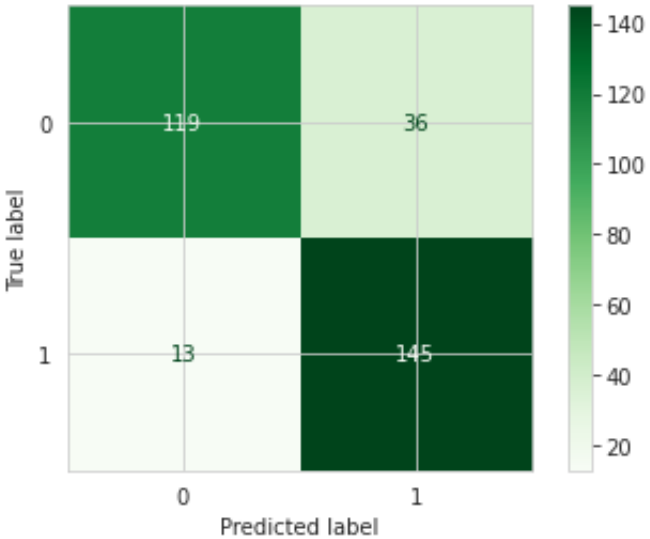


LDA

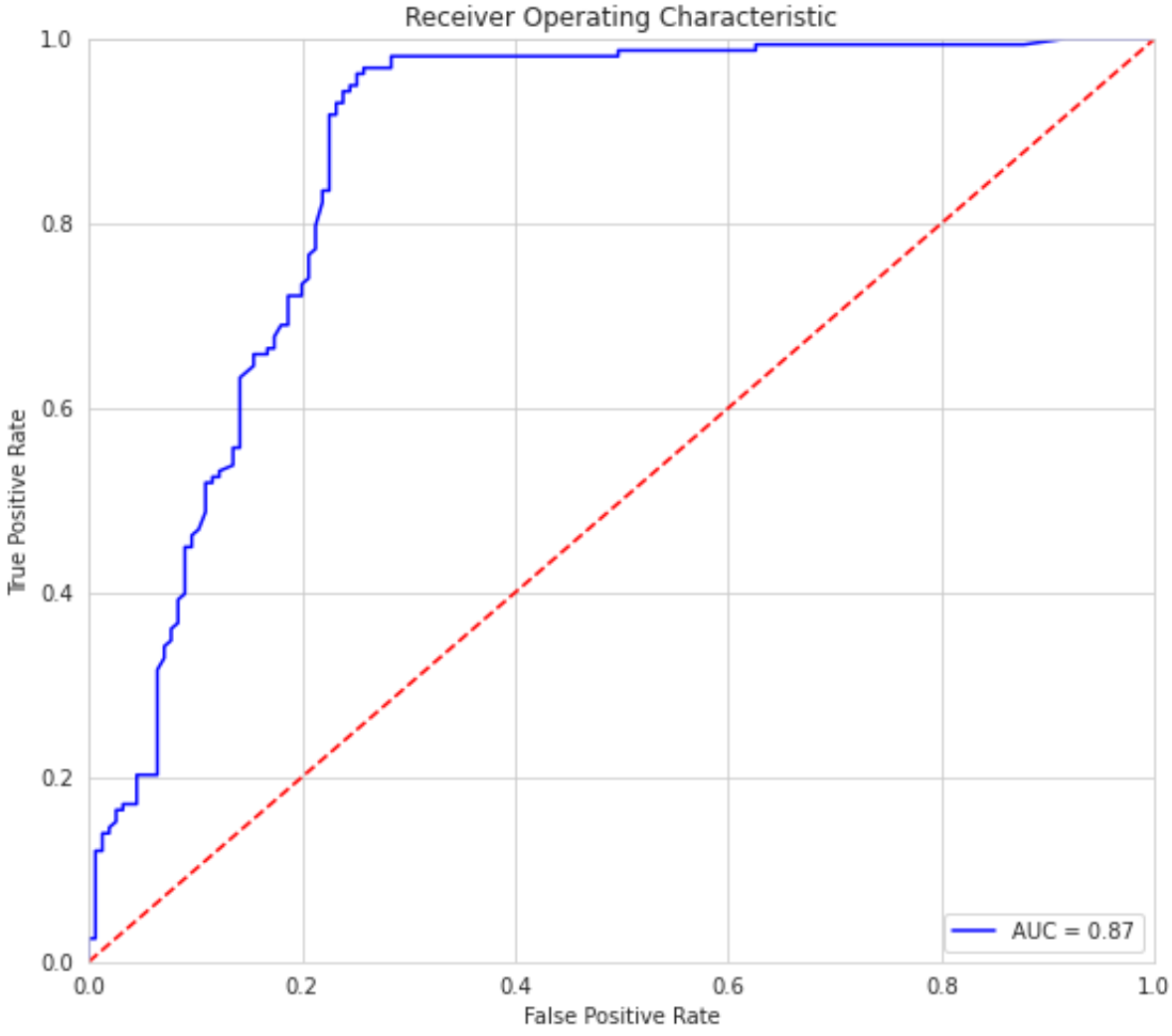
In [250]:
```
model_lda = LinearDiscriminantAnalysis()
model_lda.fit(X_train, y_train)
pred = model_nb.predict(X_test)
```

In [251]:
```
pred  = model_lda.predict(X_test)
print(classification_report(y_true = y_test, y_pred = pred))
plot_confusion_matrix(model_lda, X = X_test, y_true = y_test, cmap='Gr
accuracy_lda=accuracy_score(y_test,pred)
recall_lda=recall_score(y_test,pred)
precision_lda=precision_score(y_test,pred)
print("LDA' accuracy = ",accuracy_lda )
print("LDA's recall= ",recall_lda)
print("LDA's precision= ",precision_lda,"\n" )
```

```
              precision    recall  f1-score   support

           0       0.90      0.77      0.83       155
           1       0.80      0.92      0.86       158

    accuracy                           0.84       313
   macro avg       0.85      0.84      0.84       313
weighted avg       0.85      0.84      0.84       313

LDA' accuracy =  0.8434504792332268
LDA's recall=  0.9177215189873418
LDA's precision=  0.8011049723756906
```



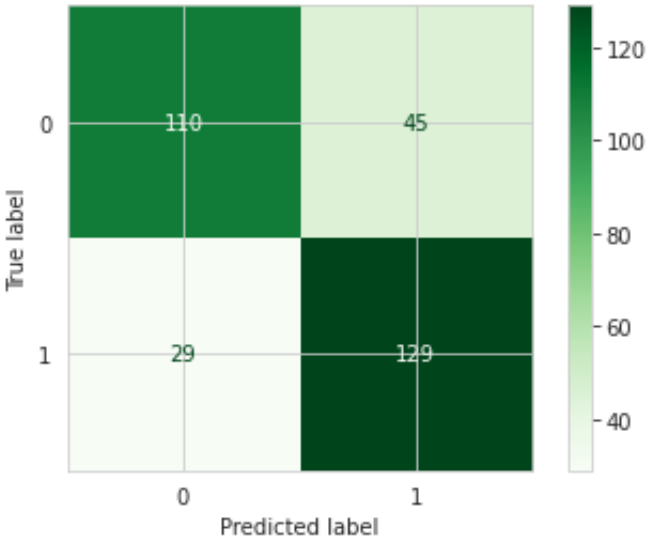
In [252]:
```
pred_prob = model_lda.predict_proba(X_test)
pred_prob = pred_prob[:,1]
metrics.roc_auc_score(y_test, pred_prob)
fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_prob)
roc_auc = metrics.auc(fpr, tpr)
plt.figure(figsize = (9,8))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



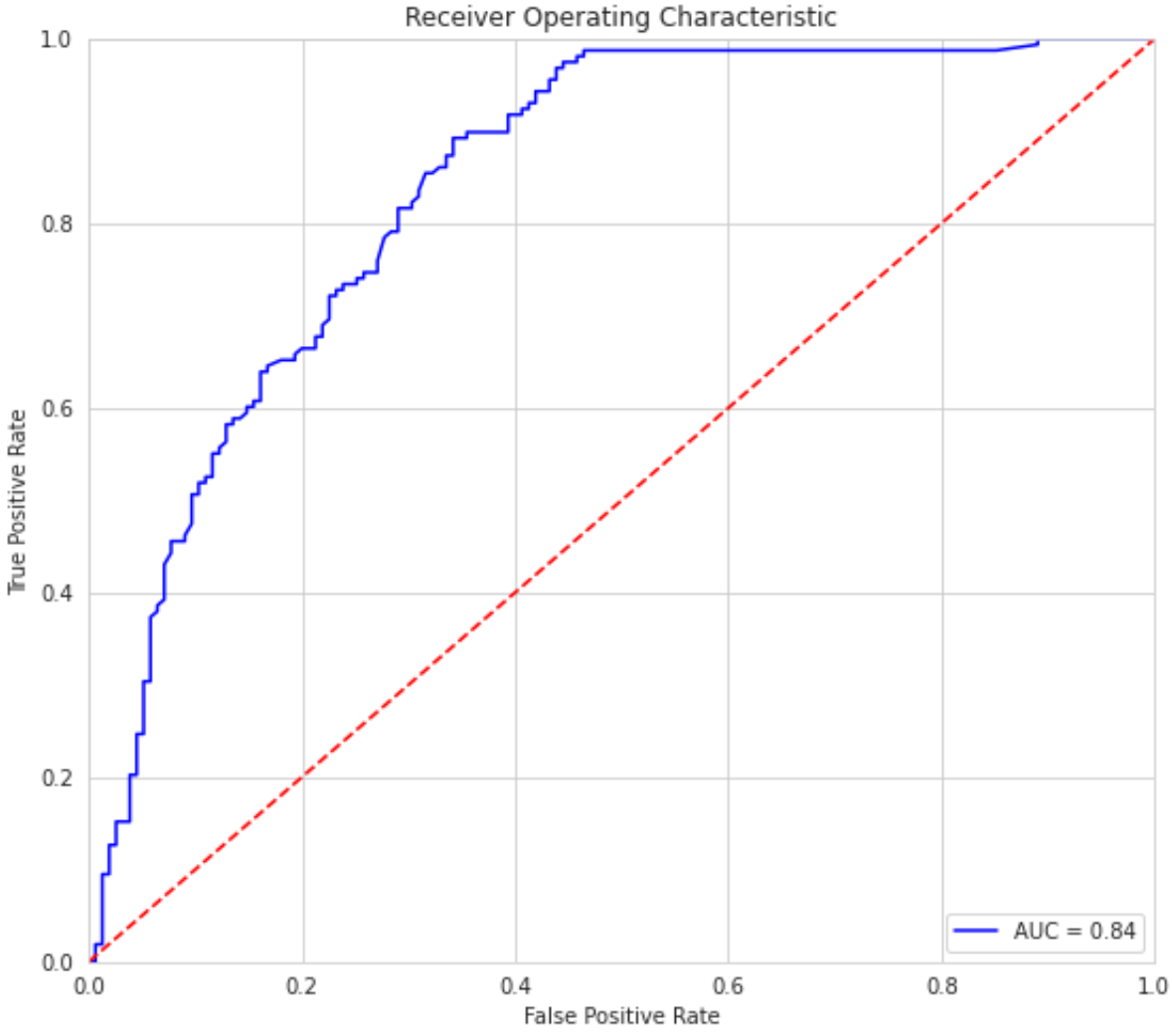
Neural Network

In [253]:
```
model_nn = MLPClassifier(hidden_layer_sizes =(150,100,50),max_iter= 30
model_nn.fit(X_train,y_train)
pred = model_nn.predict(X_test)
print(classification_report(y_true = y_test, y_pred = pred))
plot_confusion_matrix(model_nn, X = X_test, y_true = y_test, cmap='Gre
accuracy_nb=accuracy_score(y_test,pred)
recall=recall_score(y_test,pred)
precision=precision_score(y_test,pred)
print("Neural Network's accuracy = ",accuracy_nb )
print("Neural Network's recall= ",recall)
print("Neural Network's precision= ",precision,"\n" )
```

```
              precision    recall  f1-score   support

           0       0.79      0.71      0.75       155
           1       0.74      0.82      0.78       158

    accuracy                           0.76       313
   macro avg       0.77      0.76      0.76       313
weighted avg       0.77      0.76      0.76       313

Neural Network's accuracy =  0.7635782747603834
Neural Network's recall=  0.8164556962025317
Neural Network's precision=  0.7413793103448276
```



In [254]:
```
pred_prob = model_nn.predict_proba(X_test)
pred_prob = pred_prob[:,1]
metrics.roc_auc_score(y_test, pred_prob)
fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_prob)
roc_auc = metrics.auc(fpr, tpr)
plt.figure(figsize = (9,8))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Conclusion

Conclusion 1:

In this study, we analyzed and finally identified nine influential factors that most affect the mental state of employees. In terms of importance, they are： work_interfere, family history, care options, Gender, obs consequence, benefits, leave, mental health consequence, anonymity. And we analyzed how we can better improve employee well-being, how to encourage employees to actively seek mental treatment when necessary.

conclusion 2:

Six different algorithms were used to predict the mental state of future employees and to predict whether they would need mental treatment such as counseling.

ID	Model Name	Accuracy	Recall	Precision
1	Random Forest	79.55%	81.65%	78.66%
2	XGBoost	79.55%	83.54%	77.65%
3	Naive Bayes	78.27%	79.75%	77.78%
4	LDA	84.35%	91.77%	80.11%
5	KNN	80.51%	87.34%	77.09%
6	Neural Network	76.36%	81.65%	74.14%

By comparing the accuracy of the models, we found the best model:Linear Discriminant Analysis(LDA), with an accuracy of 84.35%, a recall of 91.77% and precision of 80.11%, which proves that we can effectively identify 91.77% of the employees who need mental treatment. Such efficient and accurate prediction model can help companies save a lot of labor cost, help employees get better mental state, improve work efficiency, increase performance, etc.

Data Resource

https://osmihelp.org/research (https://osmihelp.org/research)