Employee_Attrition_4

October 13, 2022

Predicting and Understanding Employee Attrition

Overview

We define attrition as an employee leaving the company. This includes resignations, and all other types of terminations. A common question that many employers seek to answer is how to predict attrition before it happens, and what attributes are related to attrition. Investigating attrition can save time and money, and provide helpful insight to try and counteract attrition. The major questions we seek to answer: What variables or attributes are associated with attrition? Who might be expected to leave?

Because our goal is to create an analysis that can be understood and applied, we want to create a less flexible, simpler model so we can optimize for inference. This means we will stick to a parametric approach when possible, and leverage available domain knowledge and insights to select our variables.

In order to better understand this problem, we can use logistic regression analysis and other classification methods to identify individuals who are at risk of attrition, based on characteristics of employees who have already left. We can also leverage p-values and other statistical methods to identify the significance of coefficient values and variables, so we can attempt to answer the question of which variables are more strongly predictive of attrition. We use classification methods like logistic regression because we want to predict a discrete class label / event, attrition or no attrition.

[1]:
```python
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
import scipy.stats as stats
sn.set(rc={'figure.figsize':(6,3)})
import warnings
warnings.filterwarnings("ignore")
```

Dataset

I will be using a dataset from Kaggle, created by IBM data scientists for the express purpose of analyzing attrition for human capital analytics teams. The reason I have selected this dataset is because it is clean, and does not contain any sensitive information, as actual employee data is not be available due to security concerns.

> Link to dataset: https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

```
[2]: data = pd.read_csv("attrit_data.csv")
     data.head()
```

```
[2]:    Age Attrition     BusinessTravel  DailyRate              Department  \
     0   41       Yes      Travel_Rarely       1102                   Sales
     1   49        No  Travel_Frequently        279  Research & Development
     2   37       Yes      Travel_Rarely       1373  Research & Development
     3   33        No  Travel_Frequently       1392  Research & Development
     4   27        No      Travel_Rarely        591  Research & Development

        DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumber  \
     0                 1          2  Life Sciences              1               1
     1                 8          1  Life Sciences              1               2
     2                 2          2          Other              1               4
     3                 3          4  Life Sciences              1               5
     4                 2          1        Medical              1               7

        …  RelationshipSatisfaction StandardHours  StockOptionLevel  \
     0  …                         1            80                 0
     1  …                         4            80                 1
     2  …                         2            80                 0
     3  …                         3            80                 0
     4  …                         4            80                 1

        TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance  YearsAtCompany  \
     0                  8                      0               1               6
     1                 10                      3               3              10
     2                  7                      3               3               0
     3                  8                      3               3               8
     4                  6                      3               3               2

       YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
     0                  4                        0                     5
     1                  7                        1                     7
     2                  0                        0                     0
     3                  7                        3                     0
     4                  2                        2                     2

     [5 rows x 35 columns]
```

There are 1,470 rows (or people, in this case), 34 unique features, and one outcome variable of interest (attrition). We have a mix of categorical and continuous numeric values, and will need to investigate our predictors to ensure they are prepped and relevant to our analysis.

Data Cleaning

Part of what is great about this dataset is that it does not contain any null values. This is because it was a dataset created intentionally for modeling and analysis.

[3]:
```
data.isna().sum()
```

[3]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

The next step in data cleaning is to make sure that each of our variables have numeric representations, so we can pass them to our model. The best approach for categorical variables is to use dummy variables. Let's investigate each of our categorical variables and use dummy variables as needed.

[4]:
```
data.dtypes
```

[4]:

```
Age                          int64
Attrition                   object
BusinessTravel              object
DailyRate                    int64
Department                  object
DistanceFromHome             int64
Education                    int64
EducationField              object
EmployeeCount                int64
EmployeeNumber               int64
EnvironmentSatisfaction      int64
Gender                      object
HourlyRate                   int64
JobInvolvement               int64
JobLevel                     int64
JobRole                     object
JobSatisfaction              int64
MaritalStatus               object
MonthlyIncome                int64
MonthlyRate                  int64
NumCompaniesWorked           int64
Over18                      object
OverTime                    object
PercentSalaryHike            int64
PerformanceRating            int64
RelationshipSatisfaction     int64
StandardHours                int64
StockOptionLevel             int64
TotalWorkingYears            int64
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
dtype: object
```

[5]:

```
data_full = data.
 ↪drop(columns=['EmployeeCount','EmployeeNumber','StandardHours','Over18'])
data_full['Attrition'] = data_full['Attrition'].replace('Yes',1)
data_full['Attrition'] = data_full['Attrition'].replace('No',0)
data_full = pd.get_dummies(data_full)
```

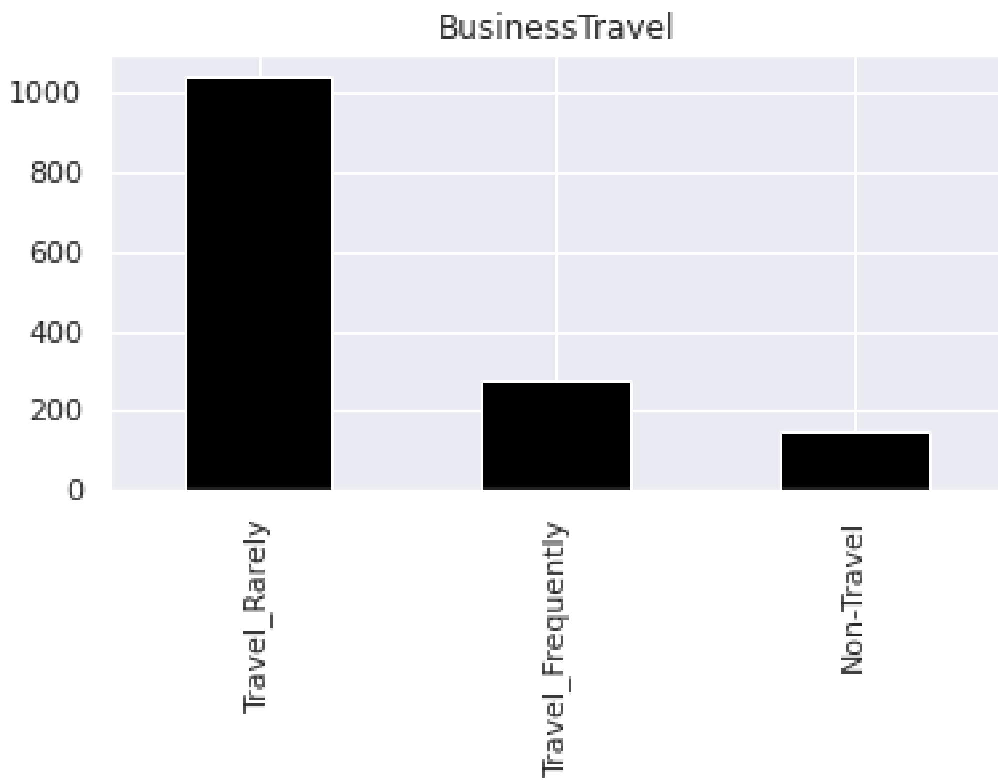
Exploratory Data Analysis

Categorical Predictors

Because we have so many predictors, before we investigate collinearity and other measures we will use to judge our predictors, we will want to understand the variance of each predictor. Because we do not know which variables will be most strongly related to our outcome, we will look for variables that have higher variance. Variables that do not have differences between groups are not as likely to provide helpful information for our model.

[6]:
```
data['BusinessTravel'].value_counts().
↪plot(kind='bar',title='BusinessTravel',color='black')
```

[6]:
```
<AxesSubplot:title={'center':'BusinessTravel'}>
```



[7]:
```
data['BusinessTravel'].groupby(data['Attrition']).describe()
```

[7]:
```
          count unique            top freq
Attrition
No         1233      3  Travel_Rarely  887
Yes         237      3  Travel_Rarely  156
```

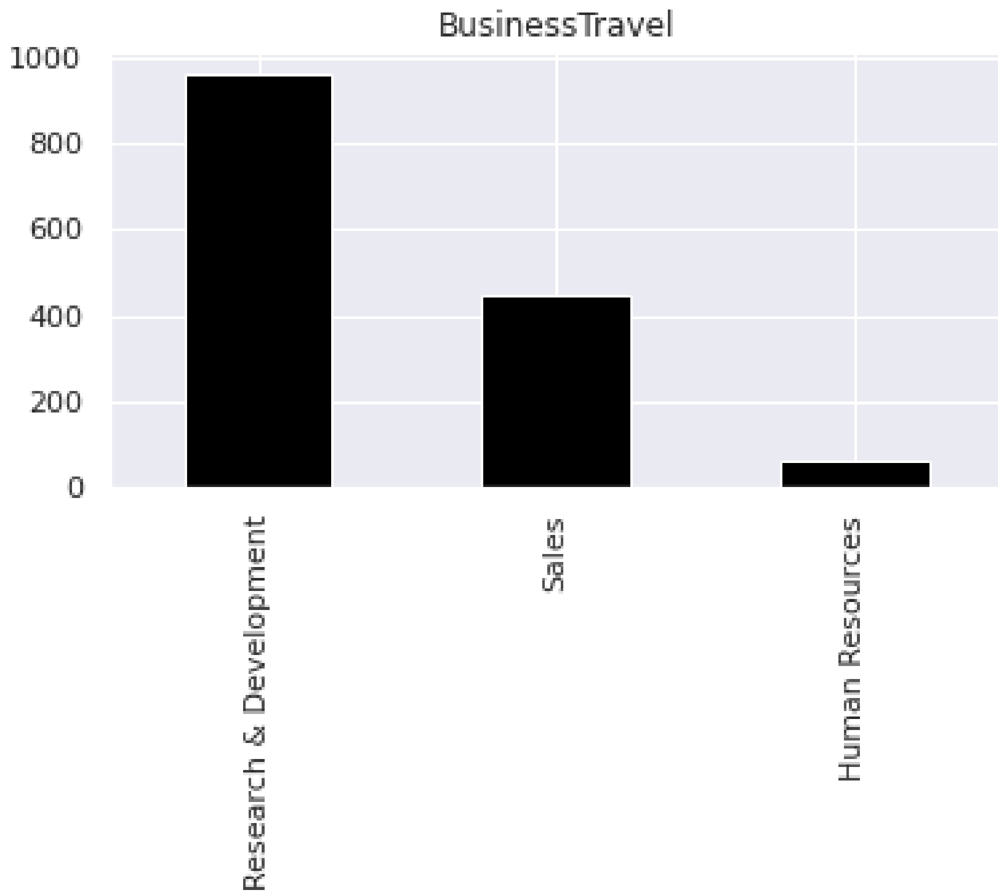
We can see that this needs to be changed to a dummy variable. We also note that we have very few individuals in the non-travel category - this may negatively impact the model if we include this variable. We will drop this variable from our analysis.

[8]:
```
data = data.drop(columns=['BusinessTravel'])
```

Department

```
[9]: data['Department'].value_counts().
       ↪plot(kind='bar',title='BusinessTravel',color='black')
```

```
[9]: <AxesSubplot:title={'center':'BusinessTravel'}>
```



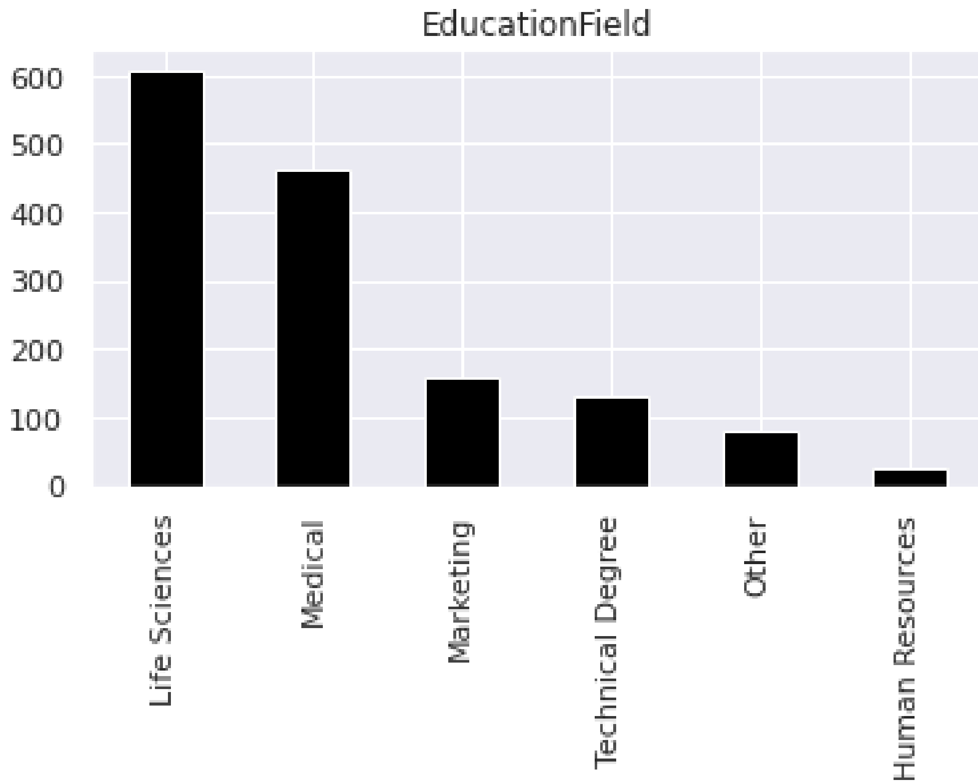
There is a department with very few individuals, Human Resources. We will drop this variable from our analysis, as this can cause convergence issues for our model.

```
[10]: data = data.drop(columns=['Department'])
```

EducationField

```
[11]: data['EducationField'].value_counts().
        ↪plot(kind='bar',title='EducationField',color='black')
```

```
[11]: <AxesSubplot:title={'center':'EducationField'}>
```



[12]:
```
data['EducationField'].groupby(data['Attrition']).value_counts()
```

[12]:
```
Attrition  EducationField
No         Life Sciences       517
           Medical             401
           Marketing           124
           Technical Degree    100
           Other                71
           Human Resources      20
Yes        Life Sciences        89
           Medical              63
           Marketing            35
           Technical Degree     32
           Other                11
           Human Resources       7
Name: EducationField, dtype: int64
```

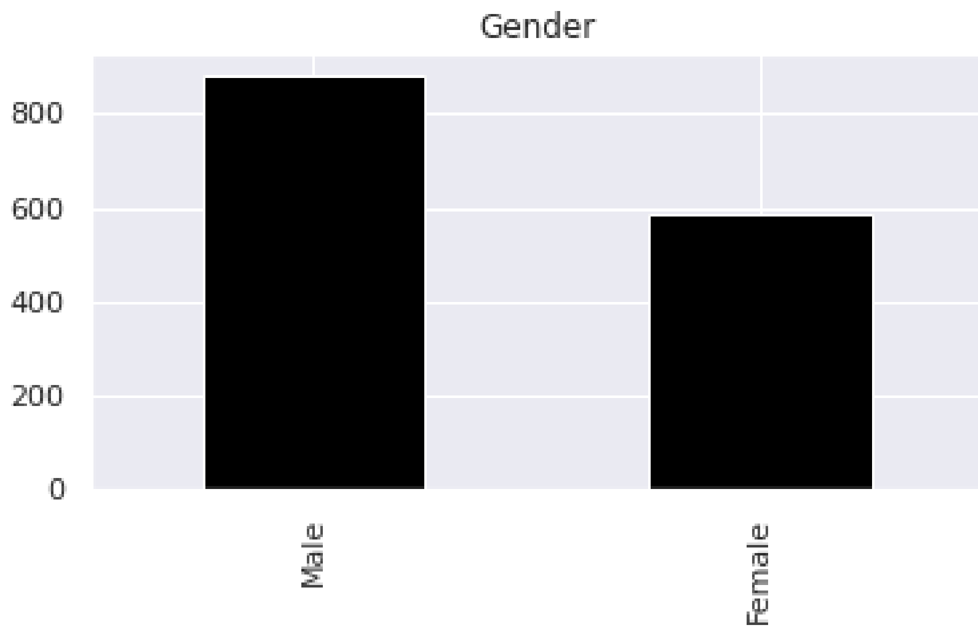
We can see that this variable has quite a few fields with very small sample sizes - this can cause convergence issues, so we will remove this data from our analysis.

[13]:
```
data = data.drop(columns=['EducationField'])
```

Gender

[14]:
```
data['Gender'].value_counts().plot(kind='bar',title='Gender',color='black')
```

[14]: <AxesSubplot:title={'center':'Gender'}>



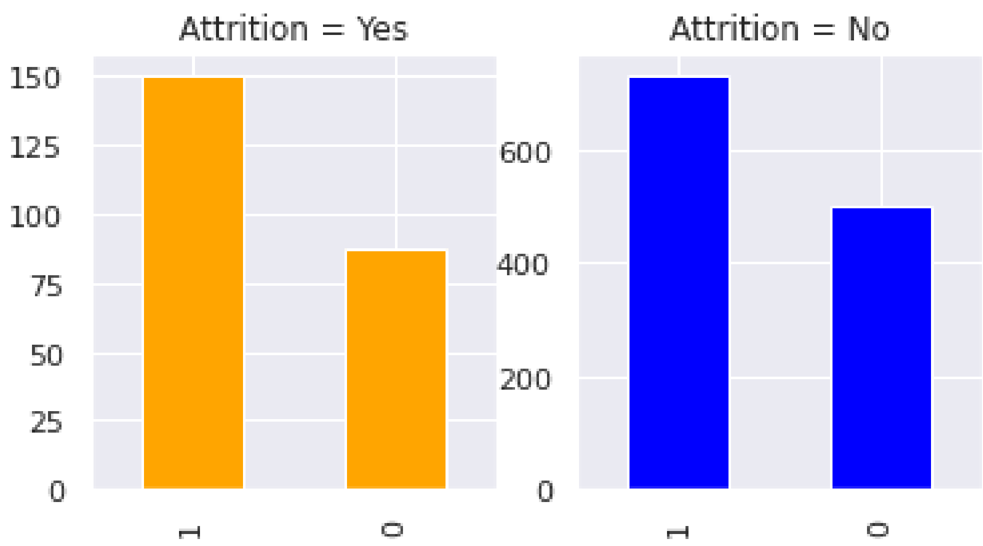
We can see that this variable can be encoded as a Boolean variable, if we change it to encode one gender. Let's encode it as male, such that GenderMale = 1 when male, GenderMale = 0 when female. In doing so, we don't need to create a dummy variable for each gender.

[15]:
```
data = data.rename(columns={'Gender':'GenderMale'})
data['GenderMale'] = data['GenderMale'].replace('Male',1)
data['GenderMale'] = data['GenderMale'].replace('Female',0)
```

Let's see whether there's a difference between attrition outcome groups in their distribution amongst the Gender variable.

[16]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 'Yes']
data_attrit['GenderMale'].value_counts().plot(kind='bar',title='Attrition =␣
↪Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 'No']
data_stay['GenderMale'].value_counts().plot(kind='bar',title='Attrition =␣
↪No',color='blue',ax=axes[1])
```

[16]: <AxesSubplot:title={'center':'Attrition = No'}>



[17]:
```
data['GenderMale'].groupby(data['Attrition']).describe()
```

[17]:

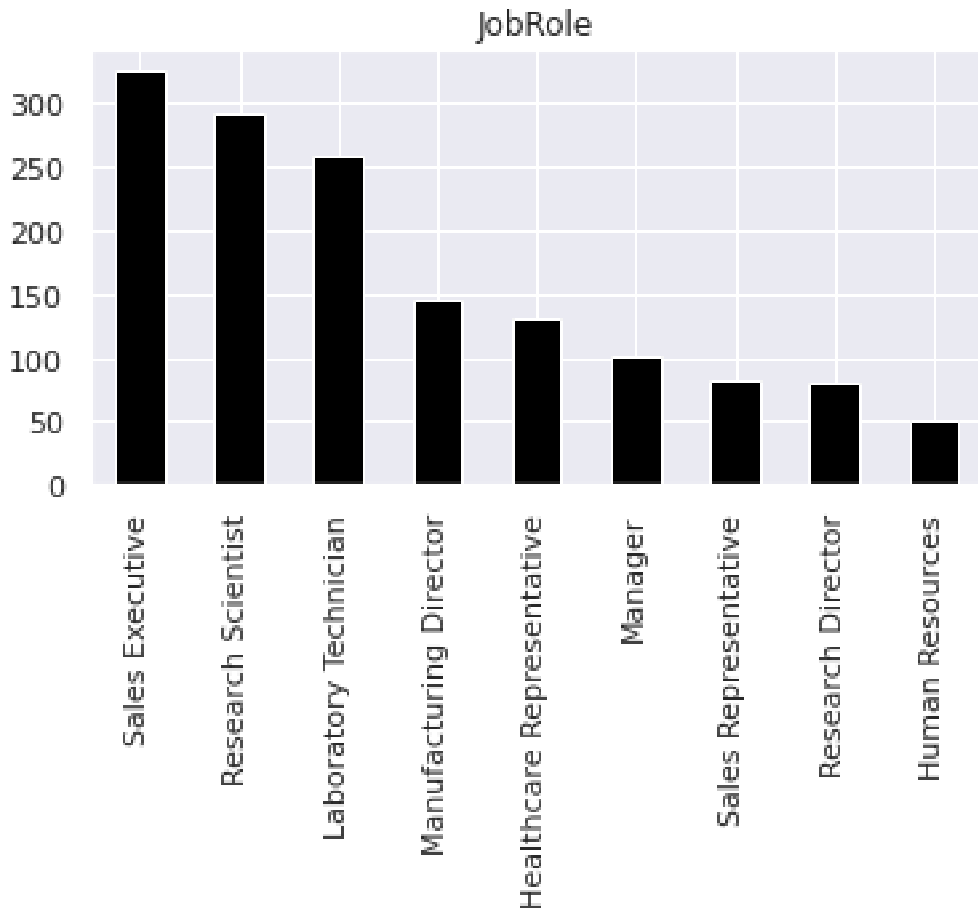
	count	mean	std	min	25%	50%	75%	max
Attrition								
No	1233.0	0.593674	0.491346	0.0	0.0	1.0	1.0	1.0
Yes	237.0	0.632911	0.483031	0.0	0.0	1.0	1.0	1.0

We won't be able to perform robust statistical measures on this variable to understand whether there is a significant difference between these two groups in terms of their distribution in the Gender variable, but by inspection, we notice that there does appear to be a difference. Variables with higher overall variance are more likely to be stronger predictors, so this provides good evidence that it is worth investigating further.

JobRole

[18]:
```
data['JobRole'].value_counts().plot(kind='bar',title='JobRole',color='black')
```

[18]: <AxesSubplot:title={'center':'JobRole'}>



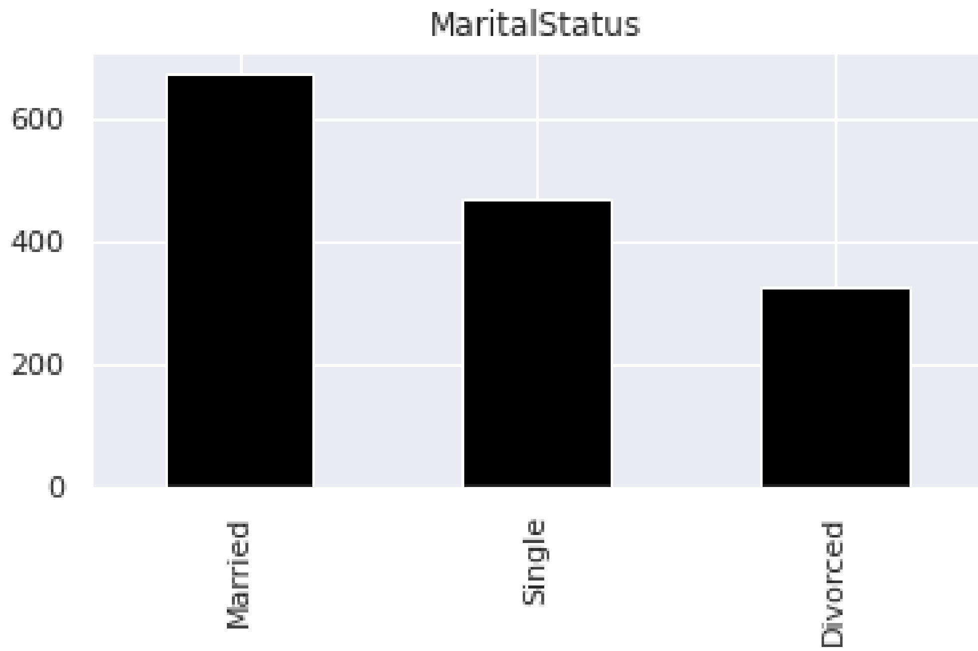
Once again, we can see that there are very few individuals in certain job roles - we will drop this variable from our analysis, as it can cause convergence issues when fitting the model.

[19]:
```
data = data.drop(columns=['JobRole'])
```

MaritalStatus

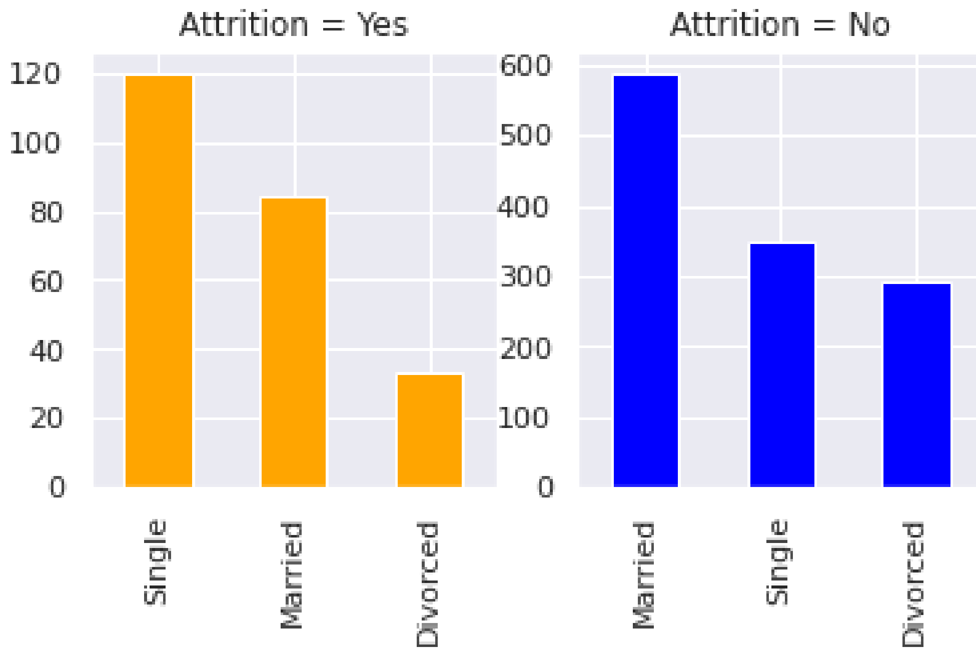
[20]:
```
data['MaritalStatus'].value_counts().
↪plot(kind='bar',title='MaritalStatus',color='black')
```

[20]:
```
<AxesSubplot:title={'center':'MaritalStatus'}>
```



[21]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 'Yes']
data_attrit['MaritalStatus'].value_counts().plot(kind='bar',title='Attrition =␣
↪Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 'No']
data_stay['MaritalStatus'].value_counts().plot(kind='bar',title='Attrition =␣
↪No',color='blue',ax=axes[1])
```

[21]:
```
<AxesSubplot:title={'center':'Attrition = No'}>
```

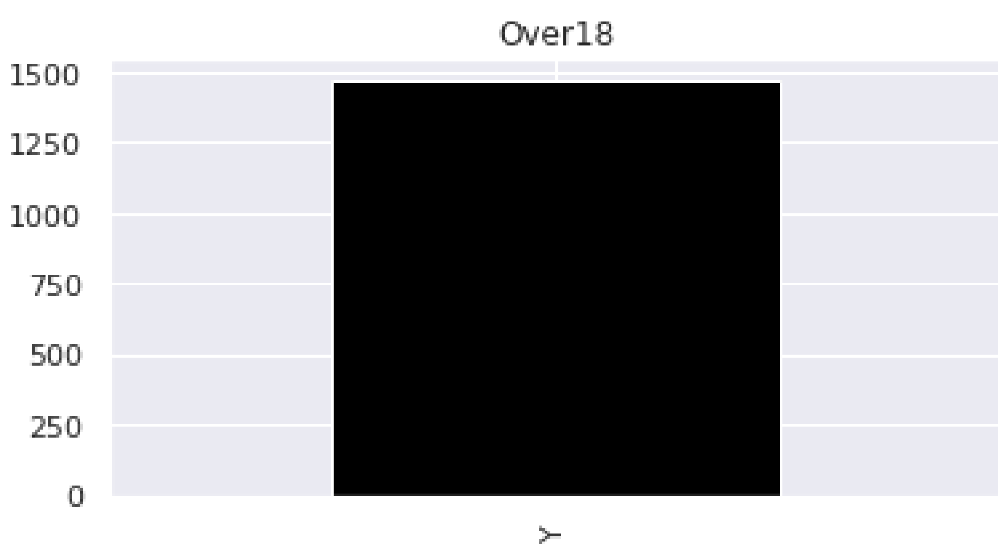


There is better distribution amongst these values than some of the other categorical variables we've investigated. This variable will need to be encoded as a dummy variable. We will investigate later whether this is a good predictor in terms of collinearity and other measures.

Over18

[22]:
```
data['Over18'].value_counts().plot(kind='bar',title='Over18',color='black')
```

[22]:
```
<AxesSubplot:title={'center':'Over18'}>
```



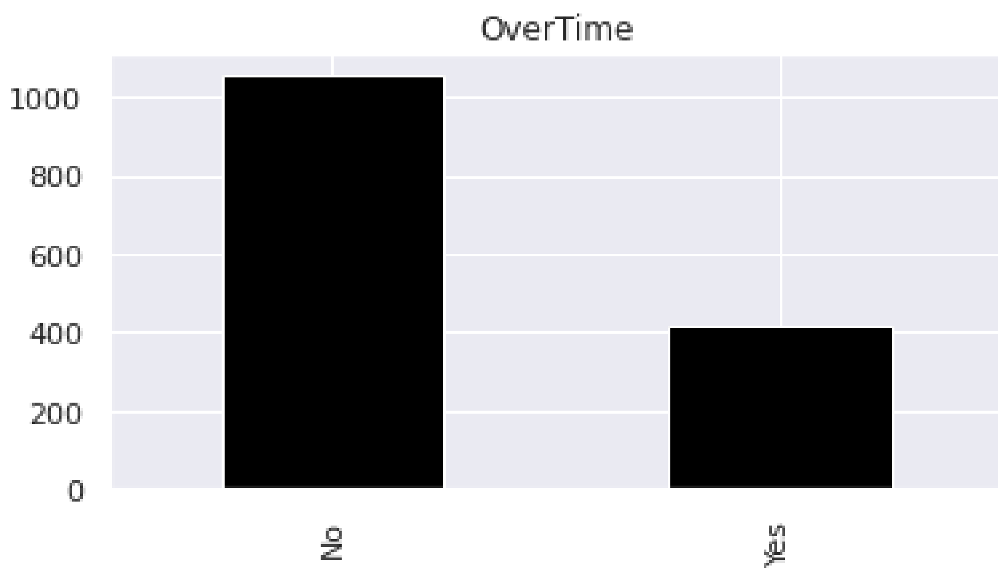
We can see that all of the samples have the same value for this predictor - let's drop it from the dataset, since it will not be helpful in our model.

[23]:
```
data = data.drop(['Over18'],axis=1)
```

OverTime

[24]:
```
data['OverTime'].value_counts().plot(kind='bar',title='OverTime',color='black')
```

[24]:
```
<AxesSubplot:title={'center':'OverTime'}>
```

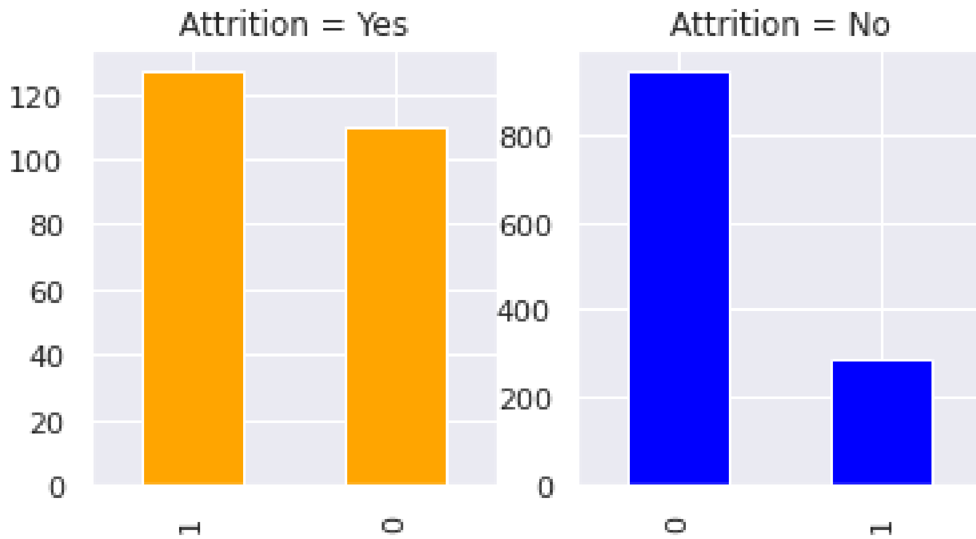


We can see that this variable can be encoded as a boolean variable.

[25]:
```
data['OverTime'] = data['OverTime'].replace('Yes',1)
data['OverTime'] = data['OverTime'].replace('No',0)
```

[26]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 'Yes']
data_attrit['OverTime'].value_counts().plot(kind='bar',title='Attrition =␣
↪Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 'No']
data_stay['OverTime'].value_counts().plot(kind='bar',title='Attrition =␣
↪No',color='blue',ax=axes[1])
```

[26]:
```
<AxesSubplot:title={'center':'Attrition = No'}>
```



We can see by inspection that there appears to be a large difference in the distribution between these two groups. Because this is a categorical variable, we won't be able to do robust statistical tests to see whether the difference is signficant, but we can look at the means of the two groups and see whether they're different.

[27]:
```
data['OverTime'].groupby(data['Attrition']).describe()
```

[27]:

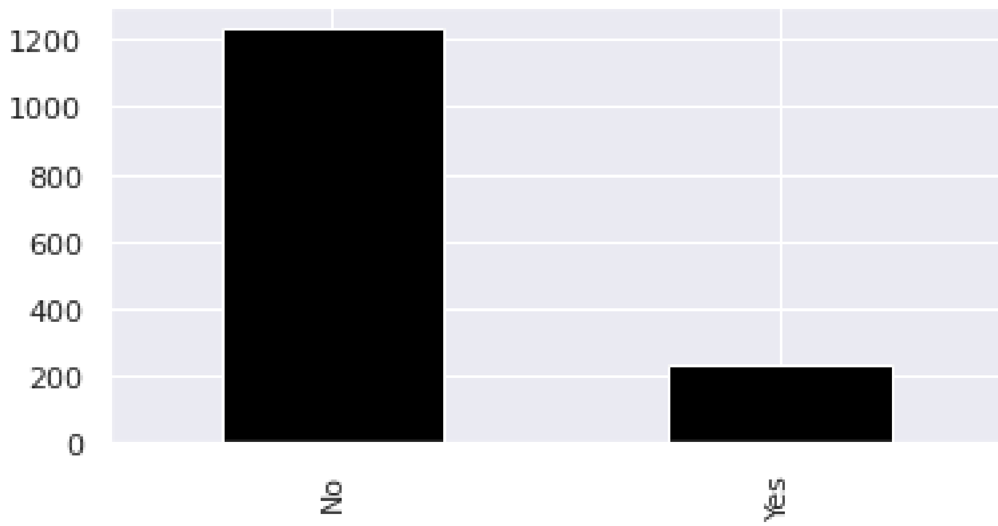
	count	mean	std	min	25%	50%	75%	max
Attrition								
No	1233.0	0.234388	0.423787	0.0	0.0	0.0	0.0	1.0
Yes	237.0	0.535865	0.499768	0.0	0.0	1.0	1.0	1.0

Attrition

Finally, we want to encode our attrition variable numerically.

[28]:
```
data['Attrition'].value_counts().plot(kind='bar',color='black')
```

[28]: <AxesSubplot:>



[29]:
```
data['Attrition'] = data['Attrition'].replace('Yes',1)
data['Attrition'] = data['Attrition'].replace('No',0)
```

Dummy Variables

Now that we've investigated all of our non-numerical variables, let's go ahead and create dummy variables for our dataset.

[30]:
```
data = pd.get_dummies(data)
data.head()
```

[30]:
```
   Age  Attrition  DailyRate  DistanceFromHome  Education  EmployeeCount  \
0   41          1       1102                 1          2              1
1   49          0        279                 8          1              1
2   37          1       1373                 2          2              1
3   33          0       1392                 3          4              1
4   27          0        591                 2          1              1

   EmployeeNumber  EnvironmentSatisfaction  GenderMale  HourlyRate  …  \
0               1                        2           0          94  …
1               2                        3           1          61  …
2               4                        4           1          92  …
3               5                        4           0          56  …
4               7                        1           1          40  …

   TotalWorkingYears  TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany  \
0                  8                      0                1               6
1                 10                      3                3              10
2                  7                      3                3               0
3                  8                      3                3               8
```

```
4                  6                      3                 3               2

   YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager  \
0                   4                        0                     5
1                   7                        1                     7
2                   0                        0                     0
3                   7                        3                     0
4                   2                        2                     2

   MaritalStatus_Divorced  MaritalStatus_Married  MaritalStatus_Single
0                       0                      0                     1
1                       0                      1                     0
2                       0                      0                     1
3                       0                      1                     0
4                       0                      1                     0

[5 rows x 32 columns]
```

Now we have 31 predictors and one outcome variable.

Let's make sure all of our variables have the correct data type (numeric) so we can continue with our exploratory data analysis.

[31]:
```
data.dtypes
```

[31]:
```
Age                         int64
Attrition                   int64
DailyRate                   int64
DistanceFromHome            int64
Education                   int64
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
GenderMale                  int64
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobSatisfaction             int64
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
OverTime                    int64
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
```

```
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
MaritalStatus_Divorced       uint8
MaritalStatus_Married        uint8
MaritalStatus_Single         uint8
dtype: object
```

Exploratory Data Analysis

Case Control Sampling

Let's take the time to understand the potential underlying distribution of our predictors and response variable. This will yield important information about what models we should use to fit the data.

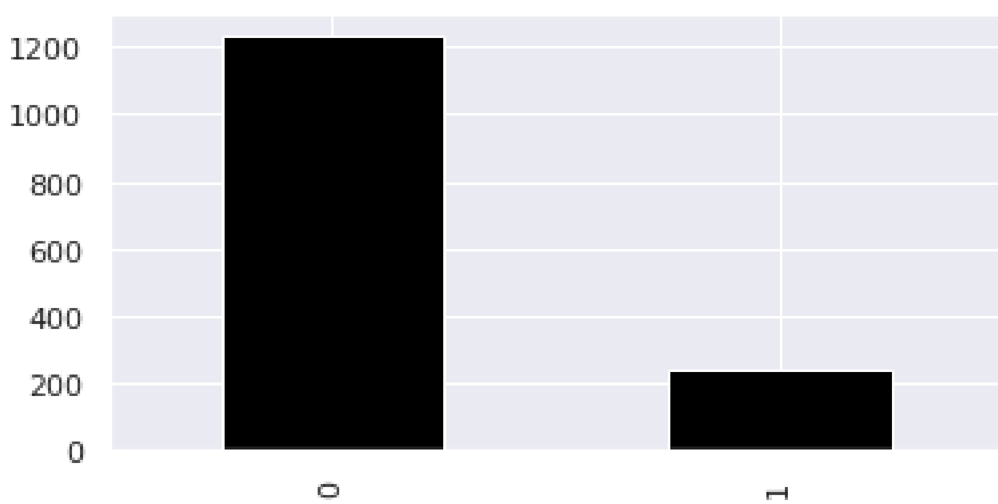
In addition, we will eliminate variables with low variance, low significance (based on our domain knowledge), and intercorrelated variables.

Response Variable: Attrition We want to encode our response variable numerically, so we can properly utilize the variable in our logistic regression modelling. Usually we do so by encoding true instances as 1 and false instances as 0.

Once this is done, let's look at the distribution of the attrition response variable.

[32]:
```python
data['Attrition'].value_counts().plot(kind='bar',color='black')
```

[32]:
```
<AxesSubplot:>
```



We can see that there is a much smaller number of cases or samples where Attrition is true. This means that our data is relatively unbalanced, and may cause issues with our modeling. Since we are going to leverage logistic regression, we will use case control sampling to address this imbalance. At the moment, we are at about 1200:200. This is a 6:1 ratio. Let's leverage case control sampling to get only roughly 800:200, so that we can reduce the variance of our parameter estimates. A good rule of thumb is to aim for a ratio of 4:1 or 5:1, since there are diminishing returns variance reduction beyond that point.

There are other methods that can be used to reduce an unbalanced dataset, such as oversampling methods like SMOTE, but they should only be done once we have completed our feature selection. Therefore, we will explore other methods later.

Exploratory Data Analysis

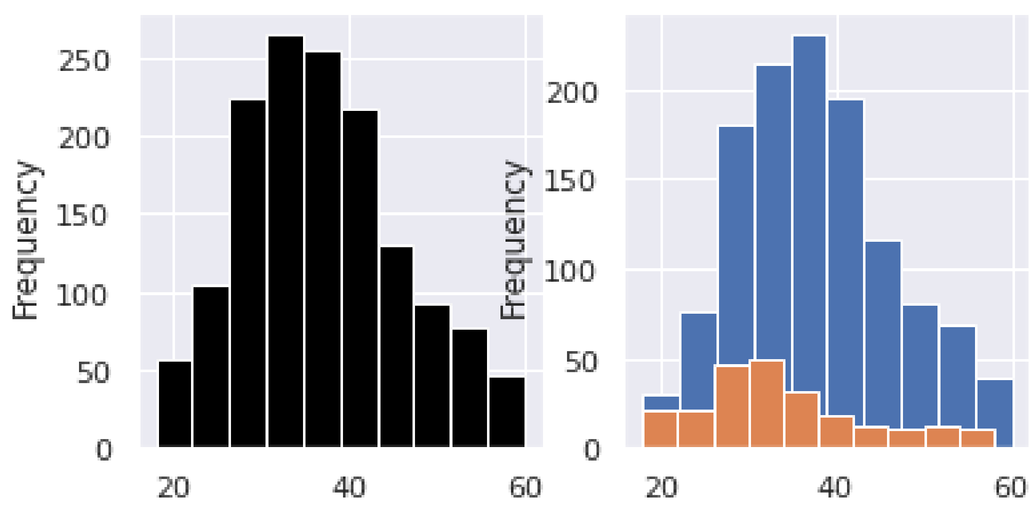
Numerical Predictors

Let's look at the distribution of each of our variables, to see if any of them can be dropped from our analysis or if there is additional information we can learn about them. We will start with our numeric variables, and use histograms to understand their distributions.

Age

[33]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['Age'].plot(ax=axes[0],kind='hist',color='black')
data['Age'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[33]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: Age, dtype: object
```



[34]:
```
data['Age'].groupby(data['Attrition']).describe()
```

[34]:

	count	mean	std	min	25%	50%	75%	max
Attrition								
0	1233.0	37.561233	8.88836	18.0	31.0	36.0	43.0	60.0
1	237.0	33.607595	9.68935	18.0	28.0	32.0	39.0	58.0

[35]:
```
attrit_age = data.query('Attrition == 1')['Age']
stay_age = data.query('Attrition == 0')['Age']
stats.levene(attrit_age, stay_age)
```

[35]:
```
LeveneResult(statistic=0.48784929772776303, pvalue=0.4849988830829256)
```

We can see that the variance between the two groups is relatively equivalent. Now we'll see whether there is a difference between the two groups in this variable. If there is a difference, we will consider including the variable in our analysis.

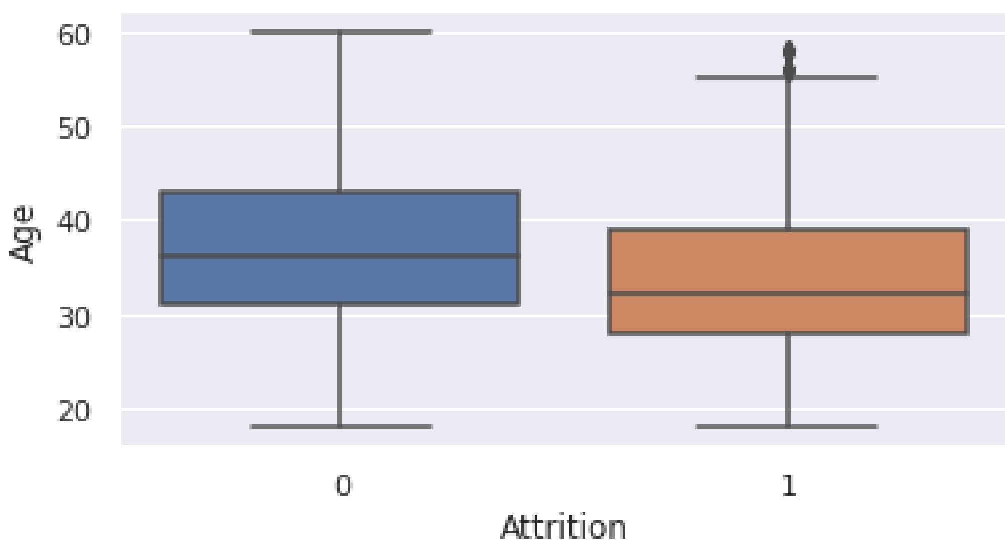
[36]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_age,stay_age,equal_var=True)
```

[36]:
```
Ttest_indResult(statistic=-6.1786638353072165, pvalue=8.356308021103649e-10)
```

We can see that the t-test results tell us that there is a significant difference between groups. For our variables, we are using the studentized t-test, since we can't necessarily assume normality. Although this variable appears to be normally distributed, for consistency, we will use the student t-test for all of our variables. This difference is visualized below:

[37]:
```
sn.boxplot(x='Attrition', y='Age', data=data)
```

[37]:
```
<AxesSubplot:xlabel='Attrition', ylabel='Age'>
```

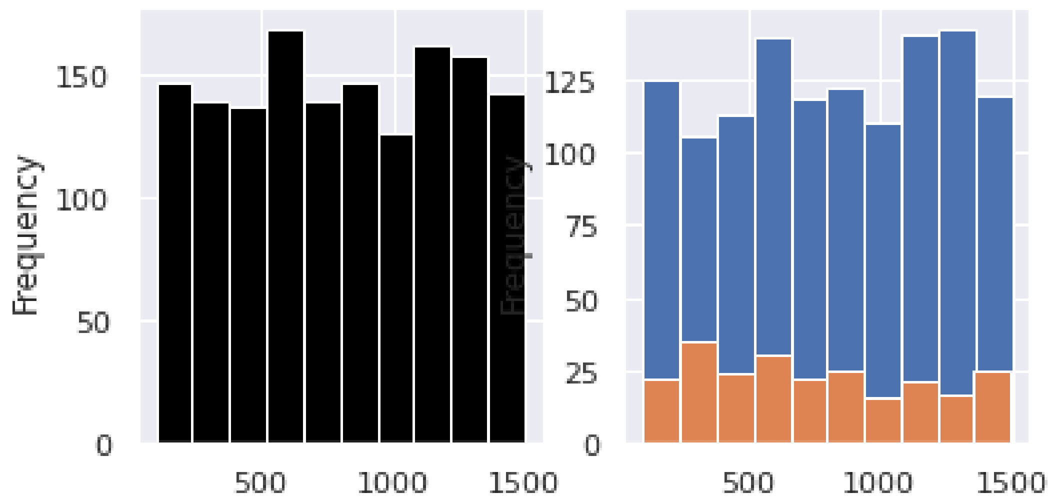


We have sufficient evidence that Age is a good predictor to include in our modeling approach.

DailyRate

[38]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['DailyRate'].plot(ax=axes[0],kind='hist',color='black')
data['DailyRate'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[38]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: DailyRate, dtype: object
```



[39]:
```
data['DailyRate'].groupby(data['Attrition']).describe()
```

[39]:

	count	mean	std	min	25%	50%	75%	max
Attrition								
0	1233.0	812.504461	403.208379	102.0	477.0	817.0	1176.0	1499.0
1	237.0	750.362869	401.899519	103.0	408.0	699.0	1092.0	1496.0

[40]:
```
attrit_rate = data.query('Attrition == 1')['DailyRate']
stay_rate = data.query('Attrition == 0')['DailyRate']
stats.levene(attrit_rate, stay_rate)
```

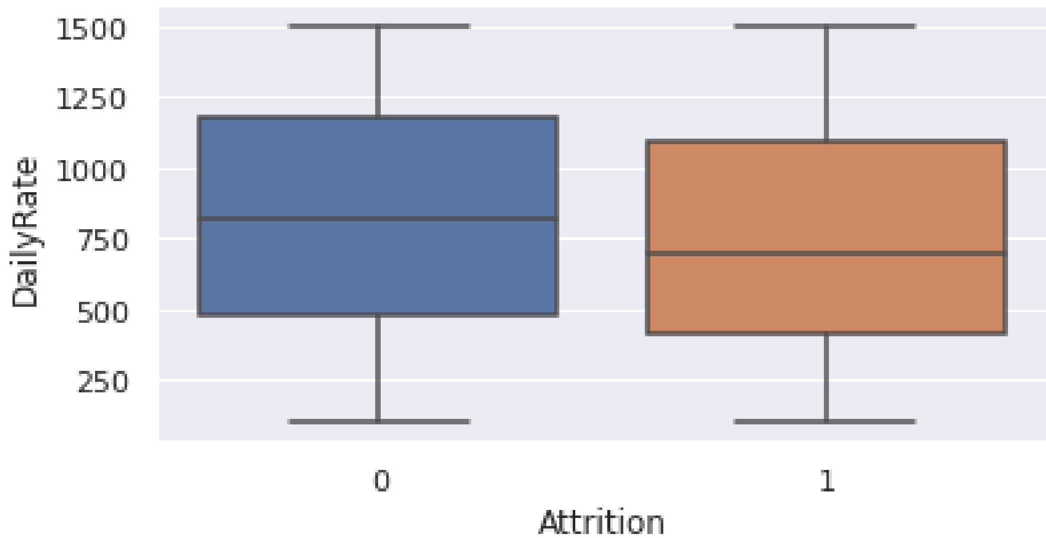
[40]:
```
LeveneResult(statistic=0.13703794064142177, pvalue=0.7112970406238526)
```

[41]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=True)
```

[41]:
```
Ttest_indResult(statistic=-2.1740836777017747, pvalue=0.02985816066026497)
```

[42]:
```
sn.boxplot(x='Attrition', y='DailyRate', data=data)
```

[42]: <AxesSubplot:xlabel='Attrition', ylabel='DailyRate'>

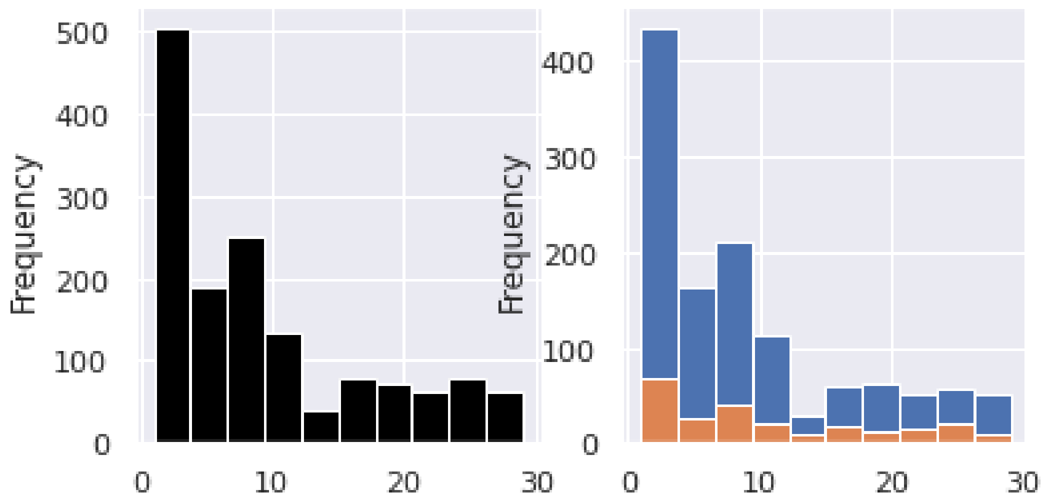


Based on our analysis, we can tell that there is a statistically significant difference between groups. This predictor might be better than others to include as a result.

DistanceFromHome

[43]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['DistanceFromHome'].plot(ax=axes[0],kind='hist',color='black')
data['DistanceFromHome'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[43]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: DistanceFromHome, dtype: object
```



[44]:
```
data['DistanceFromHome'].groupby(data['Attrition']).describe()
```

[44]:
```
             count       mean       std  min  25%  50%   75%   max
Attrition
0           1233.0   8.915653  8.012633  1.0  2.0  7.0  13.0  29.0
1            237.0  10.632911  8.452525  1.0  3.0  9.0  17.0  29.0
```

[45]:
```
attrit_dist = data.query('Attrition == 1')['DistanceFromHome']
stay_dist = data.query('Attrition == 0')['DistanceFromHome']
stats.levene(attrit_dist, stay_dist)
```

[45]: LeveneResult(statistic=3.9135257992111065, pvalue=0.04808570812266364)

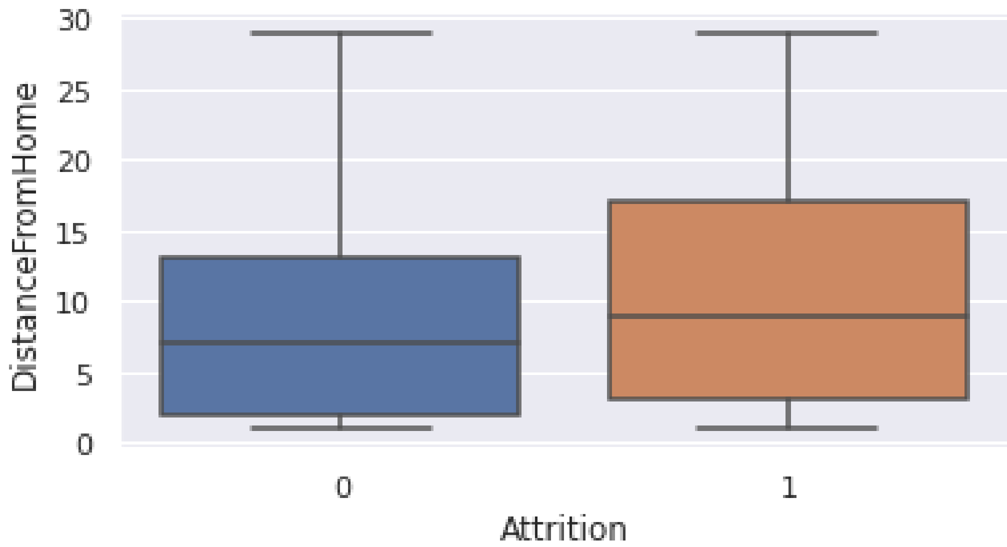
This tells us that there is strong evidence that the variance is not homogenous between groups. We will need to run a different t-test to accurately understand whether the difference between groups is significant. We will pass the 'equal_var' argument as False, since we have evidence that the variances are not equivalent between groups.

[46]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_dist,stay_dist,equal_var=False)
```

[46]: Ttest_indResult(statistic=2.888183062817627, pvalue=0.004136511971511406)

[47]:
```
sn.boxplot(x='Attrition', y='DistanceFromHome', data=data)
```

[47]: <AxesSubplot:xlabel='Attrition', ylabel='DistanceFromHome'>

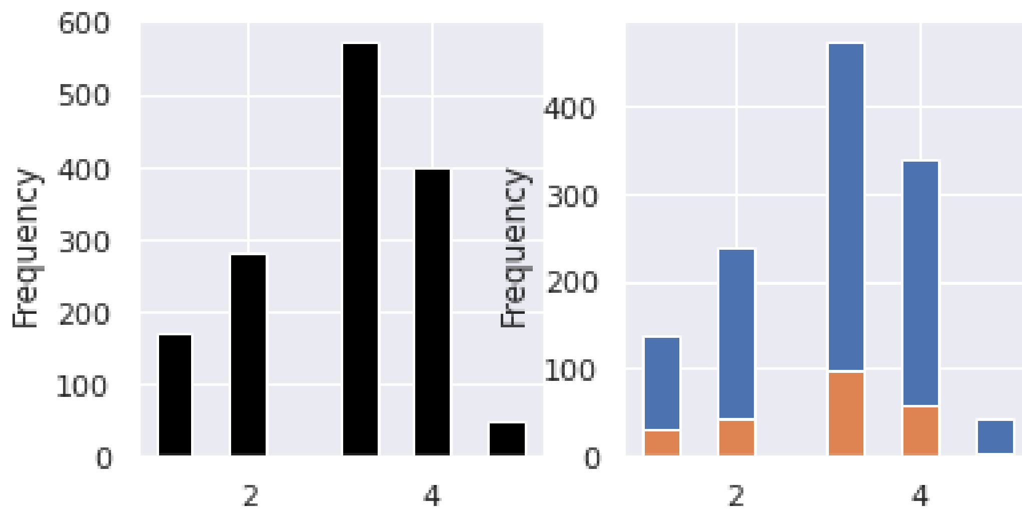


The distribution is quite skewed. However, we have strong evidence that this might be a good predictor.

Education

[48]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['Education'].plot(ax=axes[0],kind='hist',color='black')
data['Education'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[48]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: Education, dtype: object
```



This variable actually encodes a categorical scale. Education 1 'Below College' 2 'College' 3 'Bachelor' 4 'Master' 5 'Doctor'

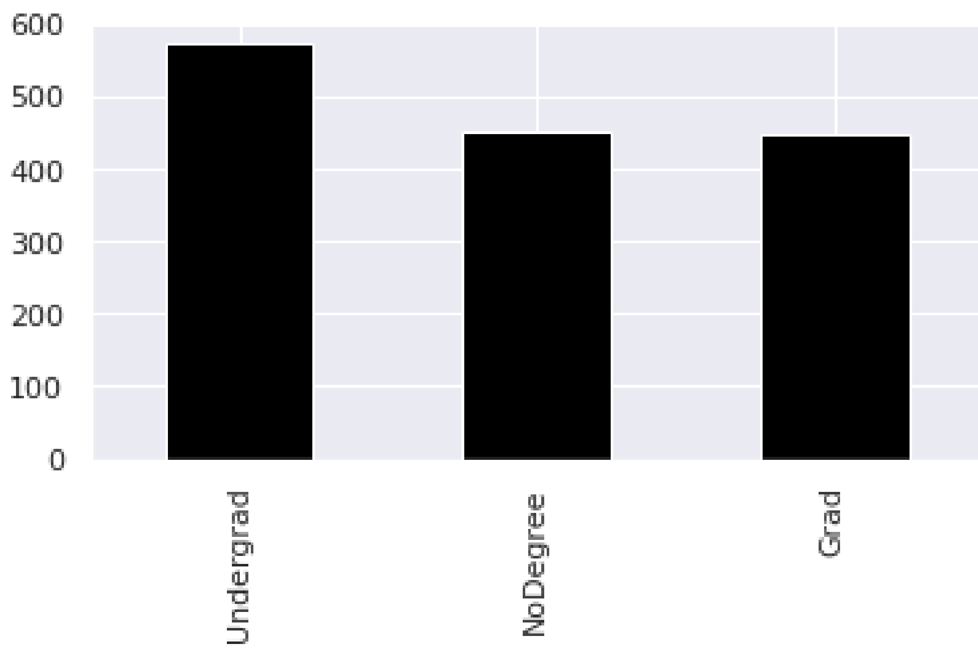
Let's create a variable to encode the highest degree level obtained, rather than using this scale. We will create the variable as follows: Education: No degree Education: undergrad Education: grad

It will align more closely with what this variable is attempting to encode. Additionally, because it is truly a categorical variable, we want to encode it correctly, so we don't experience issues later.

[49]:
```
data['Education'] = data['Education'].replace(1,'NoDegree')
data['Education'] = data['Education'].replace(2,'NoDegree')
data['Education'] = data['Education'].replace(3,'Undergrad')
data['Education'] = data['Education'].replace(4,'Grad')
data['Education'] = data['Education'].replace(5,'Grad')

data['Education'].value_counts().plot(kind='bar',color='black')
```

[49]: <AxesSubplot:>

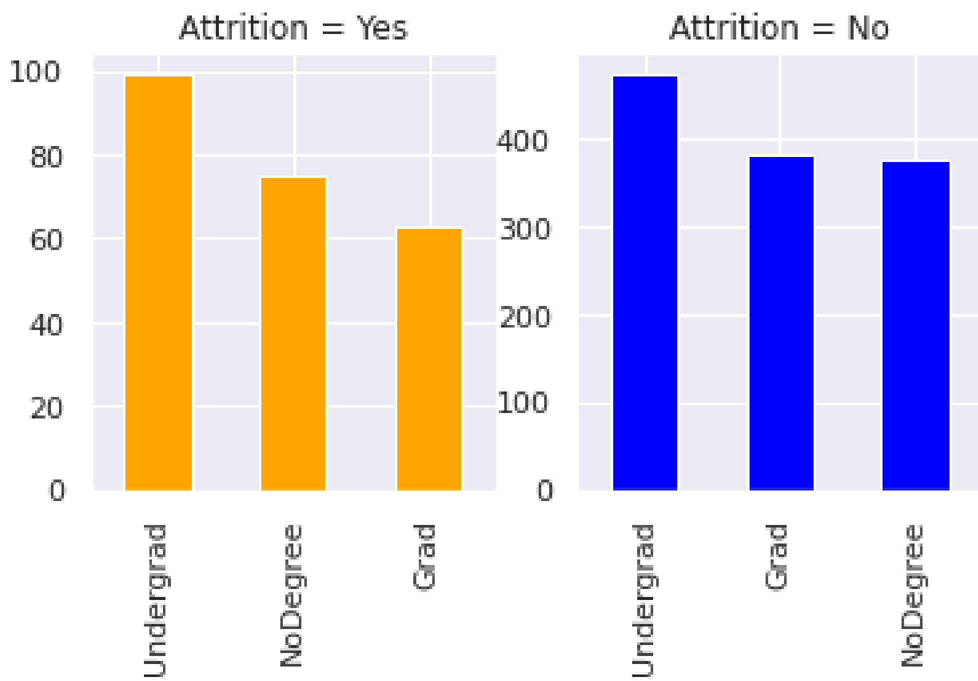


We see that our samples are more evenly distributed amongst the possible values, and that the variable more accurately captures the measured attribute.

Let's see if there's a difference between our two groups.

[50]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 1]
data_attrit['Education'].value_counts().plot(kind='bar',title='Attrition =␣
↪Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 0]
data_stay['Education'].value_counts().plot(kind='bar',title='Attrition =␣
↪No',color='blue',ax=axes[1])
```

[50]:
```
<AxesSubplot:title={'center':'Attrition = No'}>
```



We can see that there is a smaller proportion of individuals with a grad degree in attrition = yes than in attrition = no. Because this is a categorical variable, our best approach to study the variance in the predictor is by inspection.

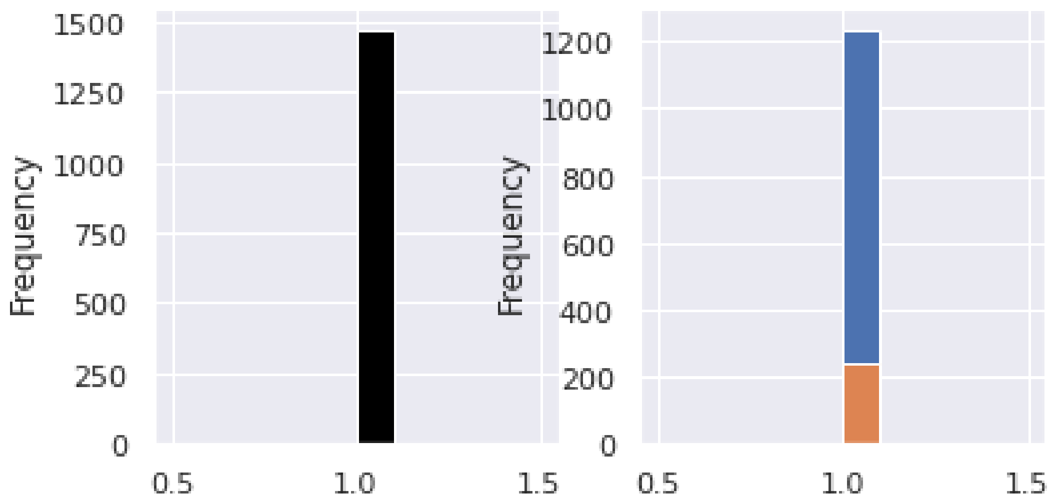
[51]:
```
data = pd.get_dummies(data)
```

EmployeeCount

[52]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['EmployeeCount'].plot(ax=axes[0],kind='hist',color='black')
data['EmployeeCount'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[52]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
```

```
Name: EmployeeCount, dtype: object
```



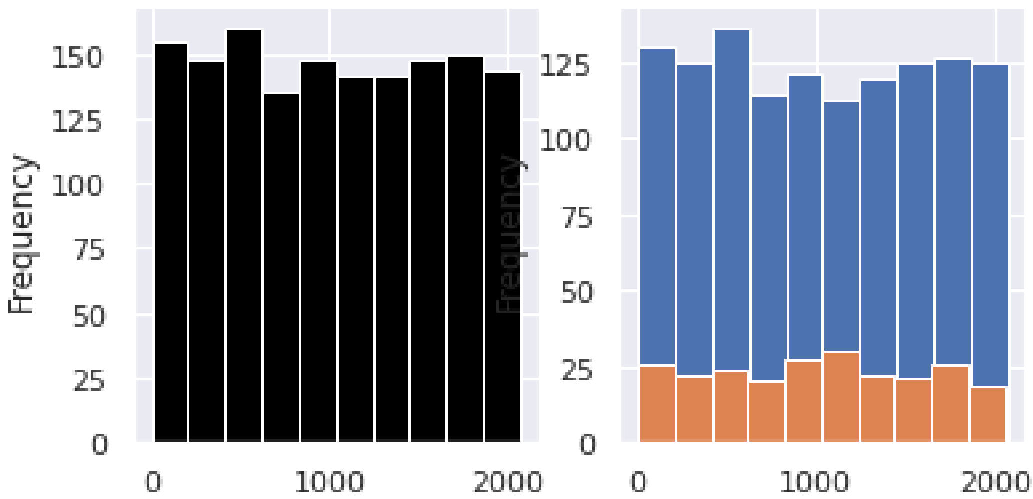
We can see that each employee has the same value for this variable - we will drop this from our analysis.

[53]:
```python
data = data.drop(columns=['EmployeeCount'])
```

EmployeeNumber

[54]:
```python
fig, axes = plt.subplots(nrows=1, ncols=2)
data['EmployeeNumber'].plot(ax=axes[0],kind='hist',color='black')
data['EmployeeNumber'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[54]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: EmployeeNumber, dtype: object
```



The histogram actually hides the fact that each employee has a unique employee number. This is common in databases, and it was likely the primary key for the table.

[55]:
```
data['EmployeeNumber'].value_counts()
```

[55]:
```
1       1
1391    1
1389    1
1387    1
1383    1
       ..
659     1
657     1
656     1
655     1
2068    1
Name: EmployeeNumber, Length: 1470, dtype: int64
```

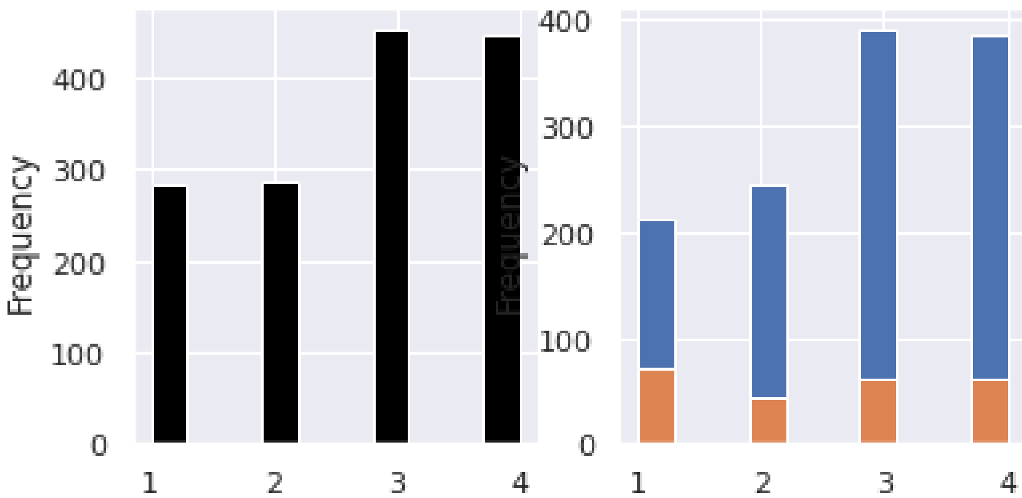
We can remove this variable from our analysis.

[56]:
```
data = data.drop(columns=['EmployeeNumber'])
```

EnvironmentSatisfaction

[57]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['EnvironmentSatisfaction'].plot(ax=axes[0],kind='hist',color='black')
data['EnvironmentSatisfaction'].groupby(data['Attrition']).
 ↪plot(ax=axes[1],kind='hist')
```

```
[57]: Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: EnvironmentSatisfaction, dtype: object
```



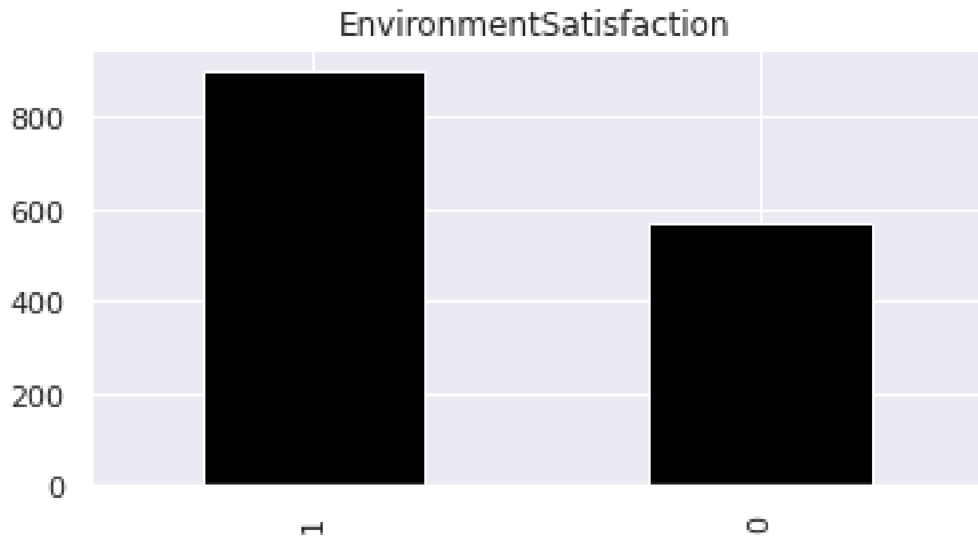
This is another variable that is truly a categorical variable. EnvironmentSatisfaction 1 'Low' 2 'Medium' 3 'High' 4 'Very High'

Let's create a boolean variable to encode this more effectively. We will create two categories: 1,2 - Less Satisfied 3,4 - MoreVery Satisfied

```
[58]: data['EnvironmentSatisfaction'] = data['EnvironmentSatisfaction'].replace(1,0)
data['EnvironmentSatisfaction'] = data['EnvironmentSatisfaction'].replace(2,0)
data['EnvironmentSatisfaction'] = data['EnvironmentSatisfaction'].replace(3,1)
data['EnvironmentSatisfaction'] = data['EnvironmentSatisfaction'].replace(4,1)

data['EnvironmentSatisfaction'].value_counts().
 ↪plot(kind='bar',title='EnvironmentSatisfaction',color='black')
```

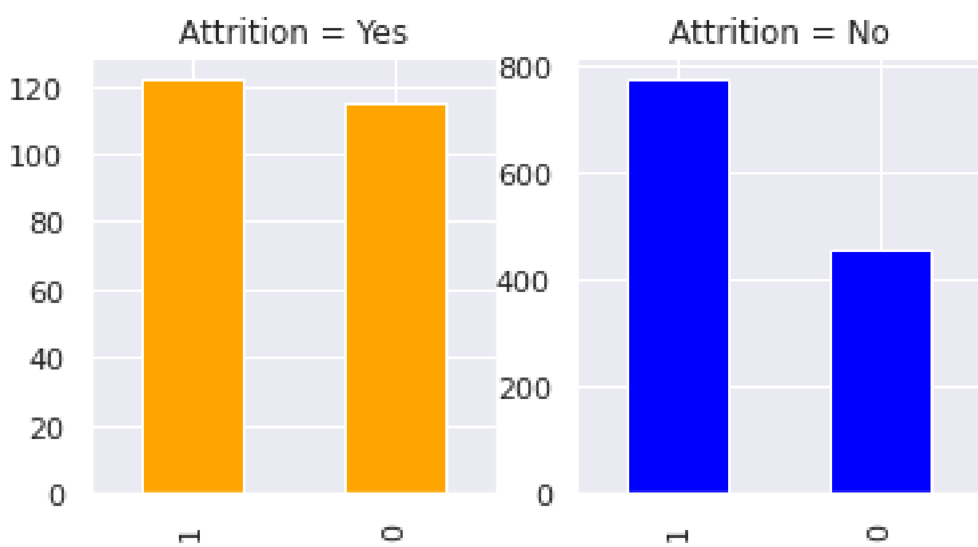
```
[58]: <AxesSubplot:title={'center':'EnvironmentSatisfaction'}>
```



Let's investigate whether there's a difference between groups.

[59]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 1]
data_attrit['EnvironmentSatisfaction'].value_counts().
    ↪plot(kind='bar',title='Attrition = Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 0]
data_stay['EnvironmentSatisfaction'].value_counts().
    ↪plot(kind='bar',title='Attrition = No',color='blue',ax=axes[1])
```

[59]:
```
<AxesSubplot:title={'center':'Attrition = No'}>
```



[60]:
```
data['EnvironmentSatisfaction'].groupby(data['Attrition']).describe()
```

[60]:

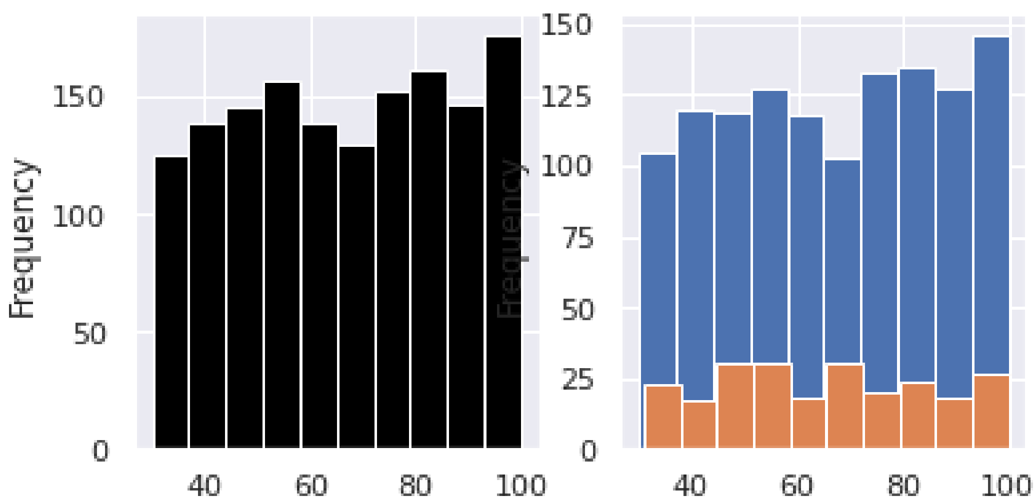
Attrition	count	mean	std	min	25%	50%	75%	max
0	1233.0	0.630170	0.482954	0.0	0.0	1.0	1.0	1.0
1	237.0	0.514768	0.500840	0.0	0.0	1.0	1.0	1.0

We can clearly see by inspection that there is a large difference between groups. This provides compelling evidence that we would like to keep this variable in our analysis.

HourlyRate

[61]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['HourlyRate'].plot(ax=axes[0],kind='hist',color='black')
data['HourlyRate'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[61]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: HourlyRate, dtype: object
```



[62]:
```
data['HourlyRate'].groupby(data['Attrition']).describe()
```

[62]:

Attrition	count	mean	std	min	25%	50%	75%	max
0	1233.0	65.952149	20.380754	30.0	48.0	66.0	83.0	100.0
1	237.0	65.573840	20.099958	31.0	50.0	66.0	84.0	100.0

[63]:
```
attrit_rate = data.query('Attrition == 1')['HourlyRate']
stay_rate = data.query('Attrition == 0')['HourlyRate']
stats.levene(attrit_rate, stay_rate)
```

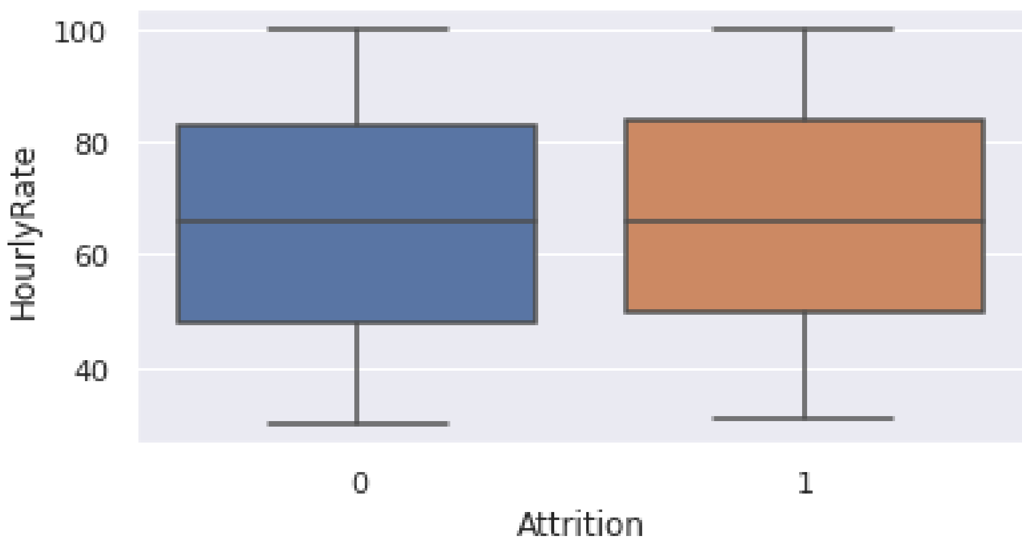
[63]:
```
LeveneResult(statistic=0.4510511131556991, pvalue=0.501941889460197)
```

[64]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=True)
```

[64]:
```
Ttest_indResult(statistic=-0.26228987349264493, pvalue=0.7931347689944243)
```

[65]:
```
sn.boxplot(x='Attrition', y='HourlyRate', data=data)
```

[65]:
```
<AxesSubplot:xlabel='Attrition', ylabel='HourlyRate'>
```



There is not a significant difference between groups for this variable. This means that there is less variance for this predictor. Because we are interested in the predictors with the highest variance, we will exclude this variable from our analysis.

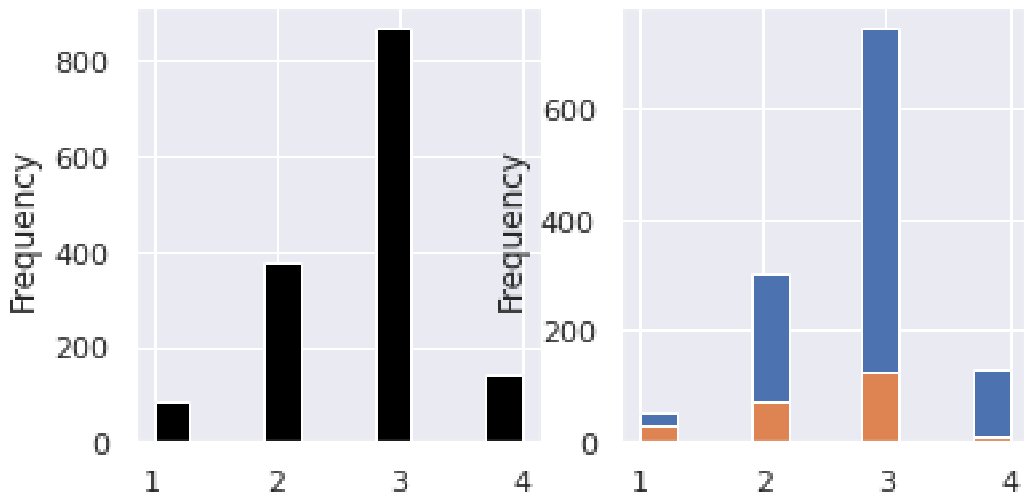
[66]:
```
data = data.drop(columns=['HourlyRate'])
```

JobInvolvement

[67]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['JobInvolvement'].plot(ax=axes[0],kind='hist',color='black')
data['JobInvolvement'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[67]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
```

```
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: JobInvolvement, dtype: object
```

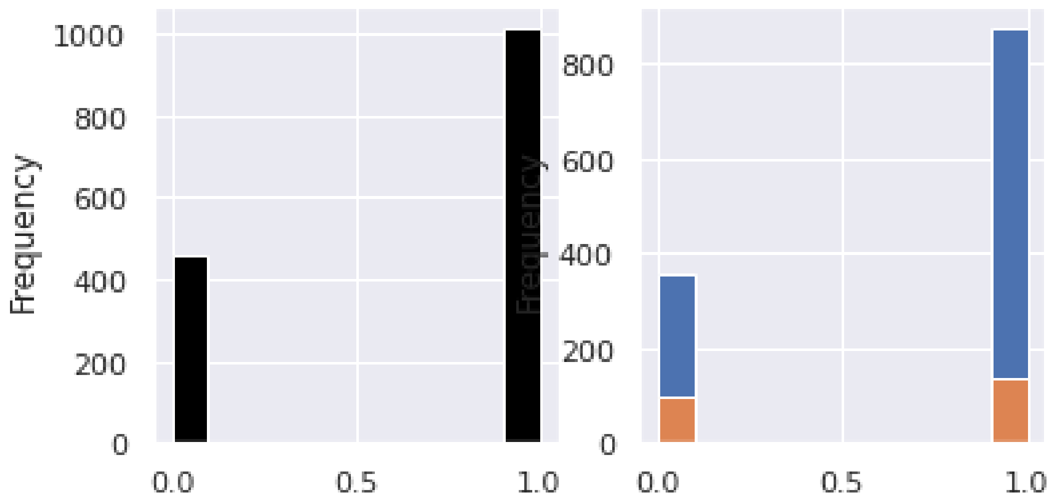


This is another variable that is truly a categorical variable. JobInvolvement 1 'Low' 2 'Medium' 3 'High' 4 'Very High' Lets create a categorical variable to encode this more effectively. We will create two categories and encode as a boolean. 1,2 - LessInvolved (0) 2,3 - MoreInvolved (1)

[68]:
```
data['JobInvolvement'] = data['JobInvolvement'].replace(1,0)
data['JobInvolvement'] = data['JobInvolvement'].replace(2,0)
data['JobInvolvement'] = data['JobInvolvement'].replace(3,1)
data['JobInvolvement'] = data['JobInvolvement'].replace(4,1)

fig, axes = plt.subplots(nrows=1, ncols=2)
data['JobInvolvement'].plot(ax=axes[0],kind='hist',color='black')
data['JobInvolvement'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

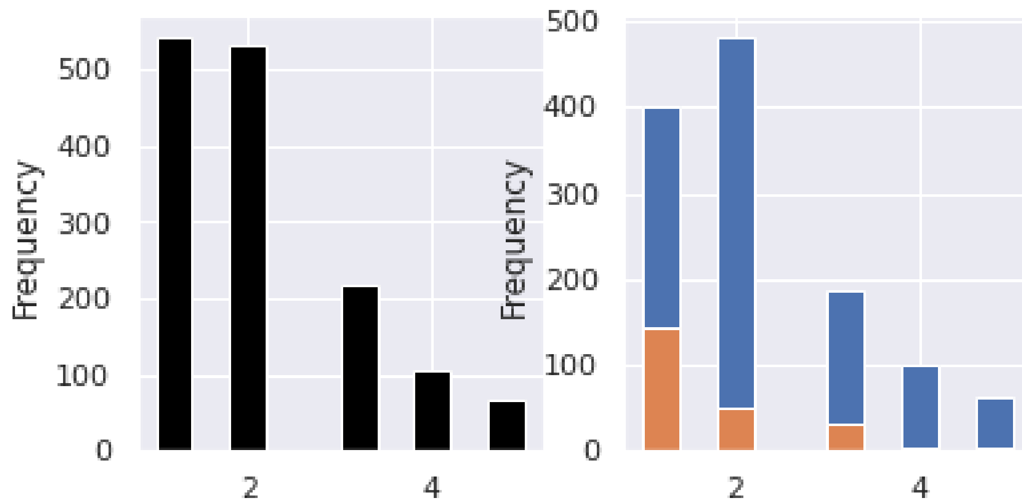
[68]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: JobInvolvement, dtype: object
```



JobLevel

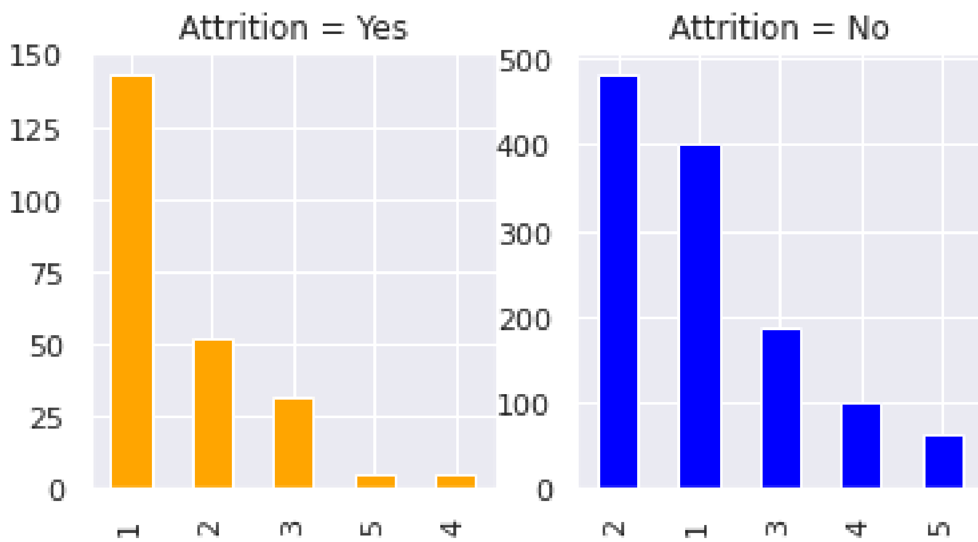
[69]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['JobLevel'].plot(ax=axes[0],kind='hist',color='black')
data['JobLevel'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[69]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: JobLevel, dtype: object
```



```
[70]: fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 1]
data_attrit['JobLevel'].value_counts().plot(kind='bar',title='Attrition =␣
 ↪Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 0]
data_stay['JobLevel'].value_counts().plot(kind='bar',title='Attrition =␣
 ↪No',color='blue',ax=axes[1])
```

```
[70]: <AxesSubplot:title={'center':'Attrition = No'}>
```

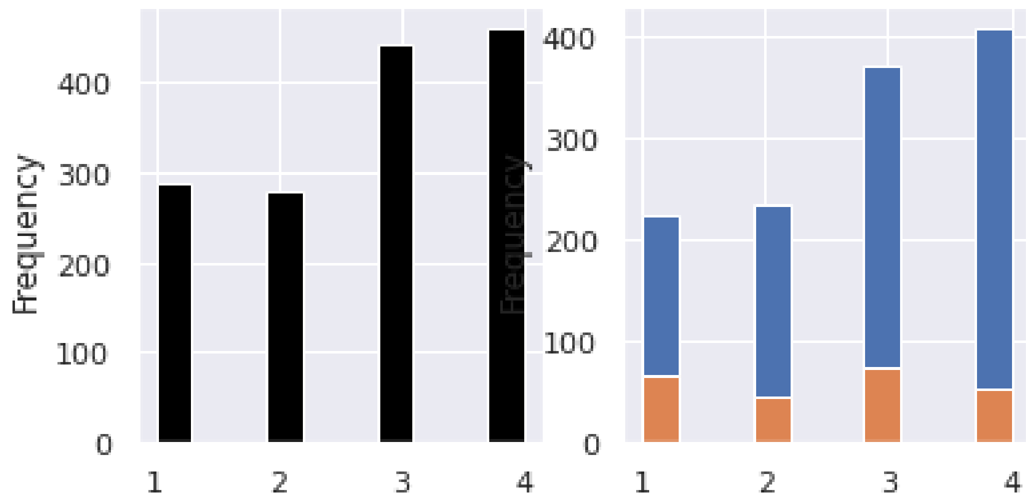


We can see that there are very few individuals with a job level above 3 in the Attrition group - this imbalance may cause issues later on. We may consider dropping this predictor from our analysis later on.

JobSatisfaction

```
[71]: fig, axes = plt.subplots(nrows=1, ncols=2)
data['JobSatisfaction'].plot(ax=axes[0],kind='hist',color='black')
data['JobSatisfaction'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

```
[71]: Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: JobSatisfaction, dtype: object
```



This is another categorical variable. We notice that this variable is similar to another variable we've looked at - environment satisfaction. We might expect that this variable is correlated to environment satisfaction.

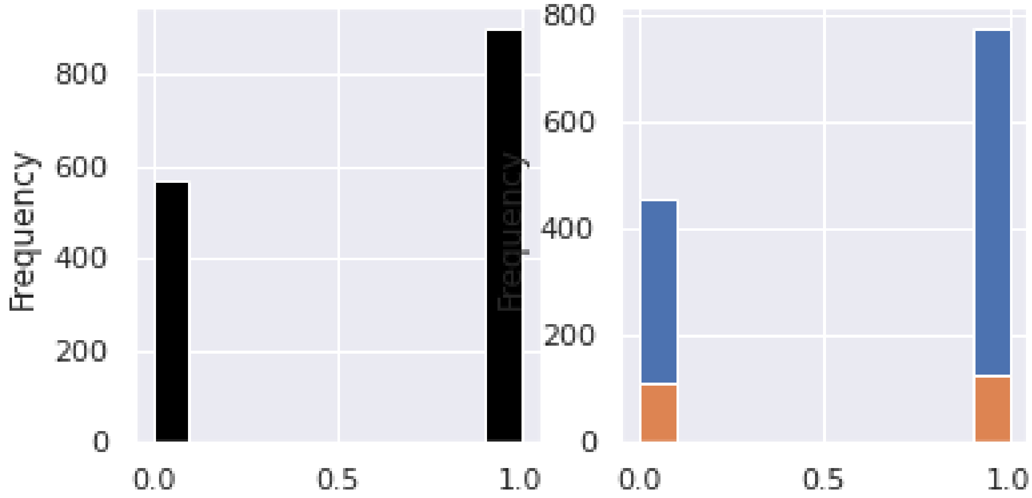
JobSatisfaction 1 'Low' 2 'Medium' 3 'High' 4 'Very High'

Let's change this variable to encode less satisfied vs. more satisfied.

[72]:
```
data['JobSatisfaction'] = data['JobSatisfaction'].replace(1,0)
data['JobSatisfaction'] = data['JobSatisfaction'].replace(2,0)
data['JobSatisfaction'] = data['JobSatisfaction'].replace(3,1)
data['JobSatisfaction'] = data['JobSatisfaction'].replace(4,1)
```

[73]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['JobSatisfaction'].plot(ax=axes[0],kind='hist',color='black')
data['JobSatisfaction'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[73]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: JobSatisfaction, dtype: object
```



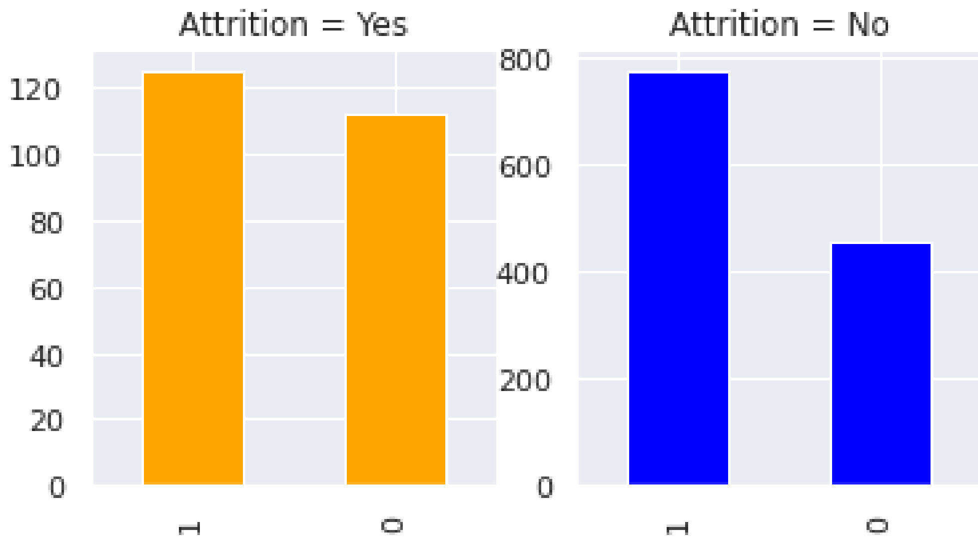
[74]:
```
data['JobSatisfaction'].groupby(data['Attrition']).describe()
```

[74]:

	count	mean	std	min	25%	50%	75%	max
Attrition								
0	1233.0	0.629359	0.483172	0.0	0.0	1.0	1.0	1.0
1	237.0	0.527426	0.500304	0.0	0.0	1.0	1.0	1.0

[75]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 1]
data_attrit['JobSatisfaction'].value_counts().plot(kind='bar',title='Attrition␣
↪= Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 0]
data_stay['JobSatisfaction'].value_counts().plot(kind='bar',title='Attrition =␣
↪No',color='blue',ax=axes[1])
```

[75]: <AxesSubplot:title={'center':'Attrition = No'}>

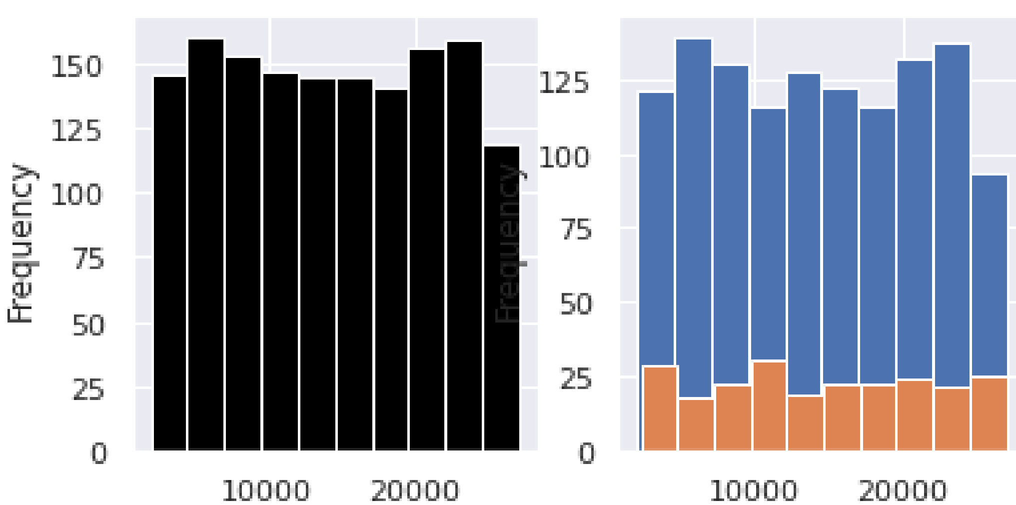


We can see by inspection that there appears to be a difference between groups.

MonthlyRate

[76]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['MonthlyRate'].plot(ax=axes[0],kind='hist',color='black')
data['MonthlyRate'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[76]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: MonthlyRate, dtype: object
```



[77]:
```
data['MonthlyRate'].groupby(data['Attrition']).describe()
```

[77]:
```
             count          mean          std     min     25%      50%  \
Attrition
0           1233.0  14265.779400  7102.260749  2094.0  7973.0  14120.0
1            237.0  14559.308017  7208.153264  2326.0  8870.0  14618.0

               75%      max
Attrition
0          20364.0  26997.0
1          21081.0  26999.0
```

[78]:
```
attrit_rate = data.query('Attrition == 1')['MonthlyRate']
stay_rate = data.query('Attrition == 0')['MonthlyRate']
stats.levene(attrit_rate, stay_rate)
```

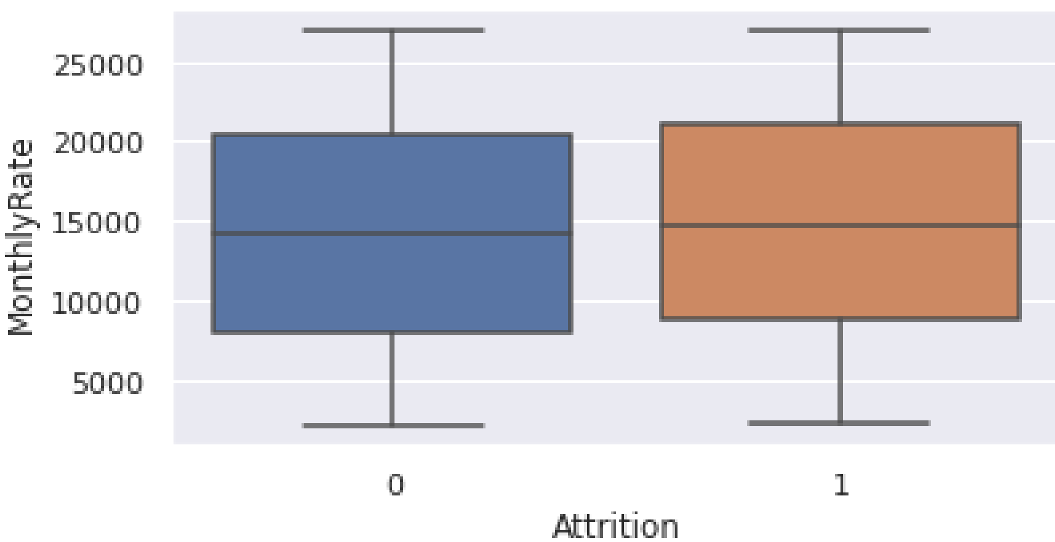
[78]:
```
LeveneResult(statistic=0.03150748458605897, pvalue=0.8591374218401133)
```

[79]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=True)
```

[79]:
```
Ttest_indResult(statistic=0.5813058211545318, pvalue=0.5611235982243015)
```

[80]:
```
sn.boxplot(x='Attrition', y='MonthlyRate', data=data)
```

[80]:
```
<AxesSubplot:xlabel='Attrition', ylabel='MonthlyRate'>
```



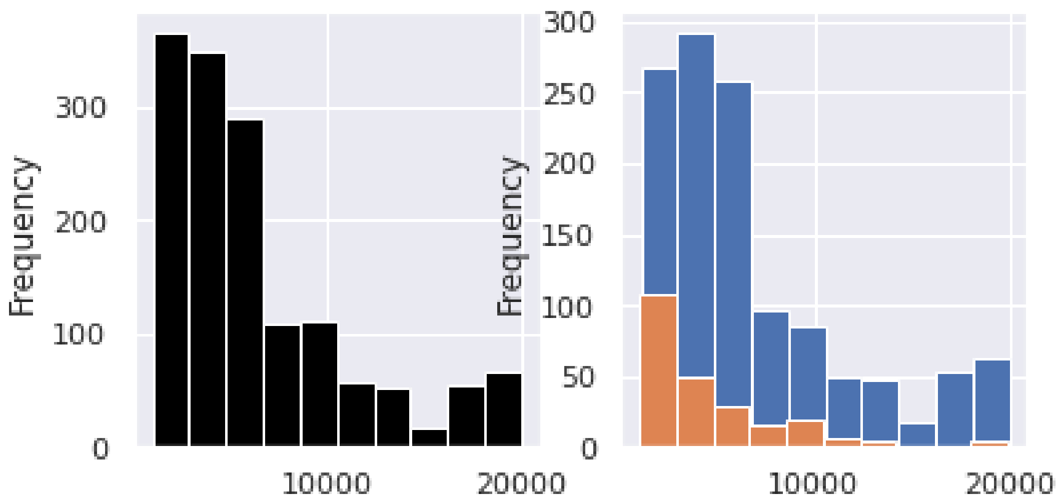
We see that there is not a significant difference between groups for this variable. We will drop this from our analysis, since we have other variables that encode income.

[81]:
```
data = data.drop(columns=['MonthlyRate'])
```

Monthly Income

[82]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['MonthlyIncome'].plot(ax=axes[0],kind='hist',color='black')
data['MonthlyIncome'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[82]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: MonthlyIncome, dtype: object
```



[83]:
```
data['MonthlyIncome'].groupby(data['Attrition']).describe()
```

[83]:
```
            count         mean          std     min     25%     50%     75%  \
Attrition
0          1233.0  6832.739659  4818.208001  1051.0  3211.0  5204.0  8834.0
1           237.0  4787.092827  3640.210367  1009.0  2373.0  3202.0  5916.0

               max
Attrition
0          19999.0
1          19859.0
```

[84]:
```
attrit_rate = data.query('Attrition == 1')['MonthlyIncome']
stay_rate = data.query('Attrition == 0')['MonthlyIncome']
stats.levene(attrit_rate, stay_rate)
```

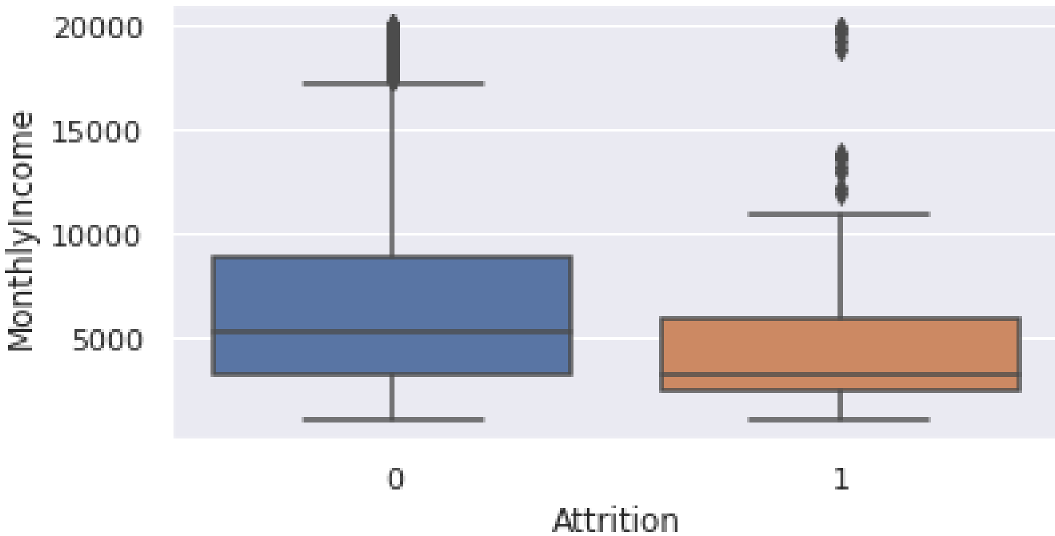
[84]:
```
LeveneResult(statistic=14.899586974568717, pvalue=0.00011830973427184532)
```

[85]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=False)
```

[85]:
```
Ttest_indResult(statistic=-7.482621586644742, pvalue=4.433588628286071e-13)
```

[86]:
```
sn.boxplot(x='Attrition', y='MonthlyIncome', data=data)
```

[86]:
```
<AxesSubplot:xlabel='Attrition', ylabel='MonthlyIncome'>
```



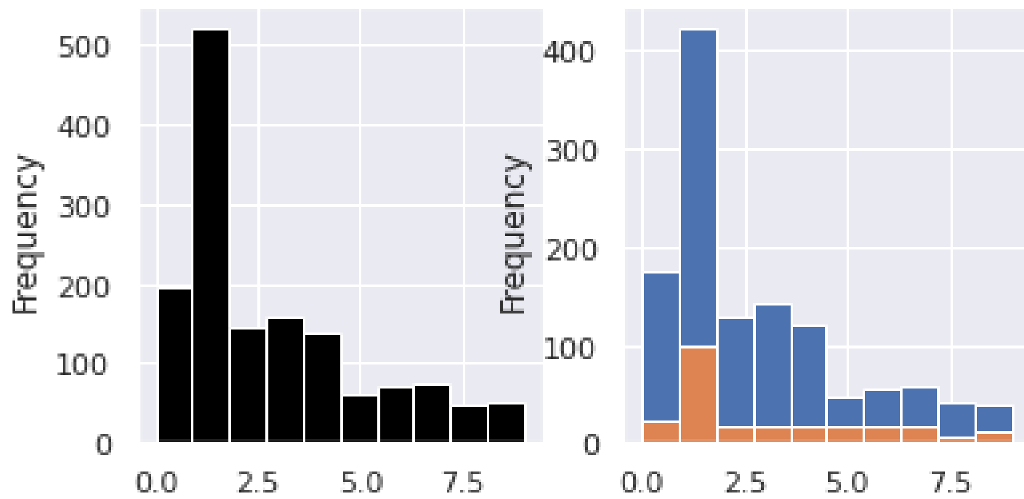
We can see that there is a significant difference between groups for this variable. However, because this variable contains a significant number of outliers, as visualized in the boxplot, and because we already have a variable that encodes income, we will drop this variable from our analysis.

[87]:
```
data = data.drop(columns=['MonthlyIncome'])
```

Num Companies Worked

[88]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['NumCompaniesWorked'].plot(ax=axes[0],kind='hist',color='black')
data['NumCompaniesWorked'].groupby(data['Attrition']).
 ↪plot(ax=axes[1],kind='hist')
```

[88]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: NumCompaniesWorked, dtype: object
```



[89]:
```python
data['NumCompaniesWorked'].groupby(data['Attrition']).describe()
```

[89]:
```
             count      mean       std  min  25%  50%  75%  max
Attrition
0           1233.0  2.645580  2.460090  0.0  1.0  2.0  4.0  9.0
1            237.0  2.940928  2.678519  0.0  1.0  1.0  5.0  9.0
```

[90]:
```python
attrit_rate = data.query('Attrition == 1')['NumCompaniesWorked']
stay_rate = data.query('Attrition == 0')['NumCompaniesWorked']
stats.levene(attrit_rate, stay_rate)
```

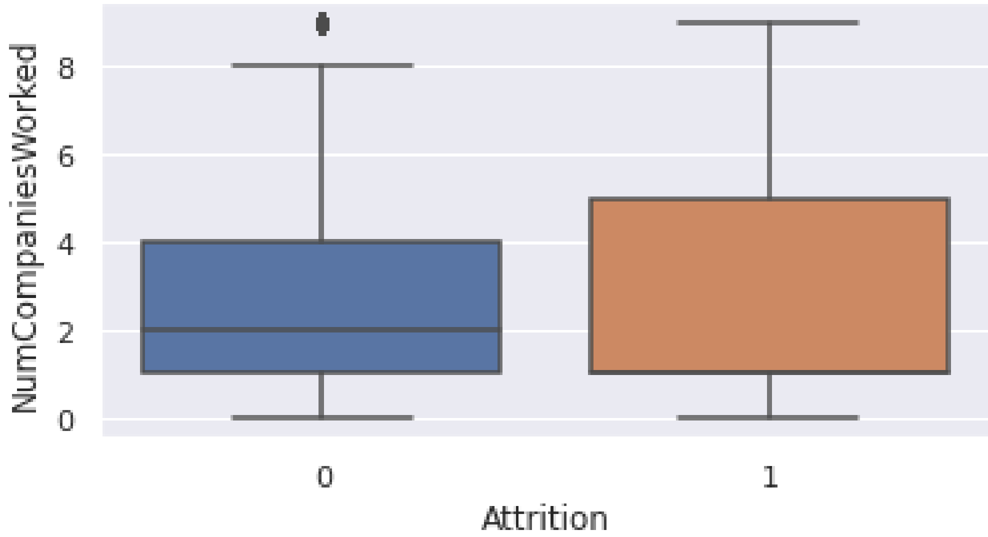
[90]:
```
LeveneResult(statistic=3.301201485128789, pvalue=0.06943305877579965)
```

[91]:
```python
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=False)
```

[91]:
```
Ttest_indResult(statistic=1.574651071928319, pvalue=0.11633402601697647)
```

[92]:
```python
sn.boxplot(x='Attrition', y='NumCompaniesWorked', data=data)
```

[92]:
```
<AxesSubplot:xlabel='Attrition', ylabel='NumCompaniesWorked'>
```

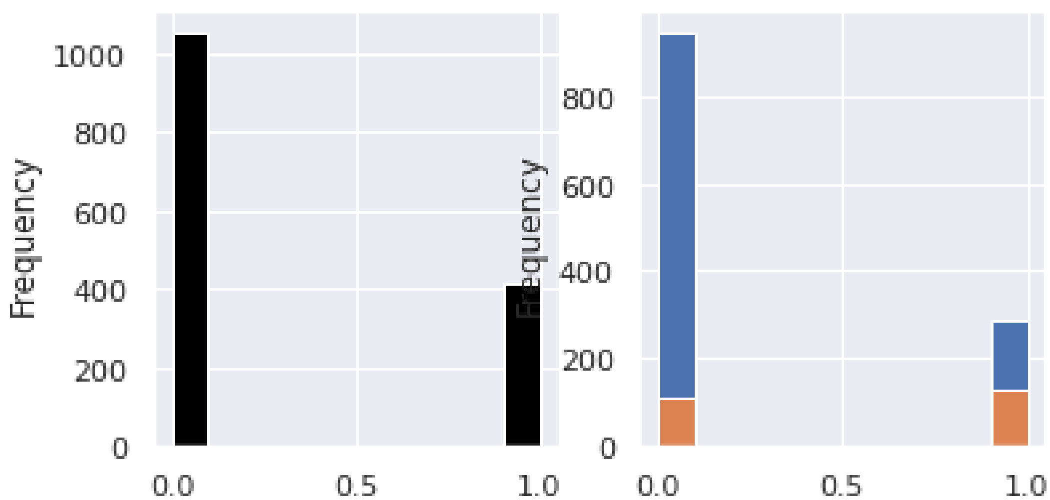


We can see that there is not a statistically significant difference between groups.

OverTime

[93]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['OverTime'].plot(ax=axes[0],kind='hist',color='black')
data['OverTime'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[93]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: OverTime, dtype: object
```



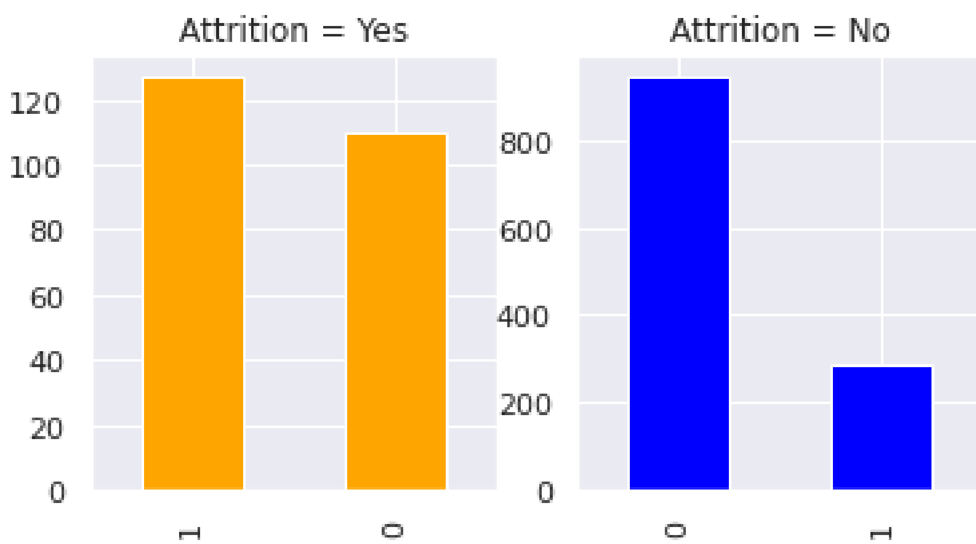
[94]:
```
data['OverTime'].groupby(data['Attrition']).describe()
```

[94]:

	count	mean	std	min	25%	50%	75%	max
Attrition								
0	1233.0	0.234388	0.423787	0.0	0.0	0.0	0.0	1.0
1	237.0	0.535865	0.499768	0.0	0.0	1.0	1.0	1.0

[95]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 1]
data_attrit['OverTime'].value_counts().plot(kind='bar',title='Attrition =␣
↪Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 0]
data_stay['OverTime'].value_counts().plot(kind='bar',title='Attrition =␣
↪No',color='blue',ax=axes[1])
```

[95]: <AxesSubplot:title={'center':'Attrition = No'}>



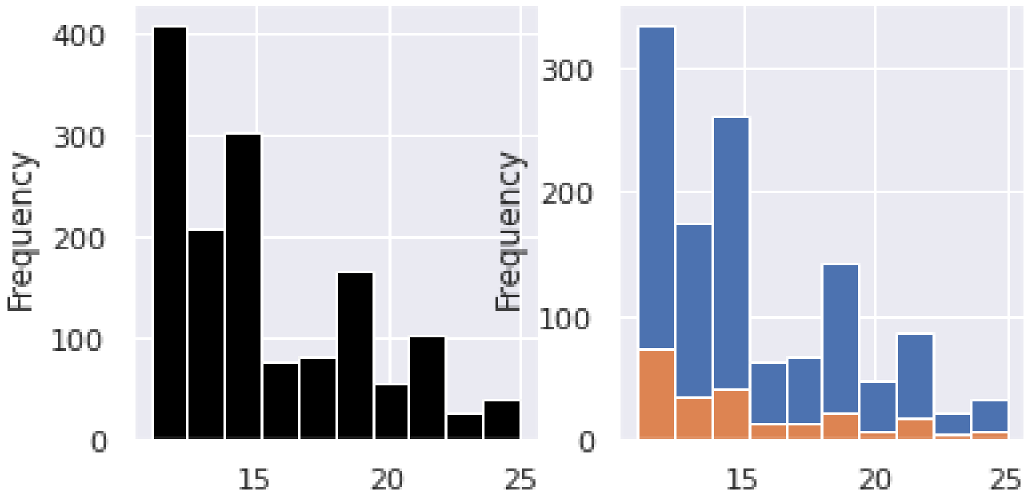
We can see by inspection that there is a strong difference between groups.

PercentSalaryHike

[96]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['PercentSalaryHike'].plot(ax=axes[0],kind='hist',color='black')
data['PercentSalaryHike'].groupby(data['Attrition']).
↪plot(ax=axes[1],kind='hist')
```

[96]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
```

```
Name: PercentSalaryHike, dtype: object
```



[97]:
```
data['PercentSalaryHike'].groupby(data['Attrition']).describe()
```

[97]:
```
             count       mean       std   min   25%   50%   75%   max
Attrition
0           1233.0  15.231144  3.639511  11.0  12.0  14.0  18.0  25.0
1            237.0  15.097046  3.770294  11.0  12.0  14.0  17.0  25.0
```

[98]:
```
attrit_rate = data.query('Attrition == 1')['PercentSalaryHike']
stay_rate = data.query('Attrition == 0')['PercentSalaryHike']
stats.levene(attrit_rate, stay_rate)
```

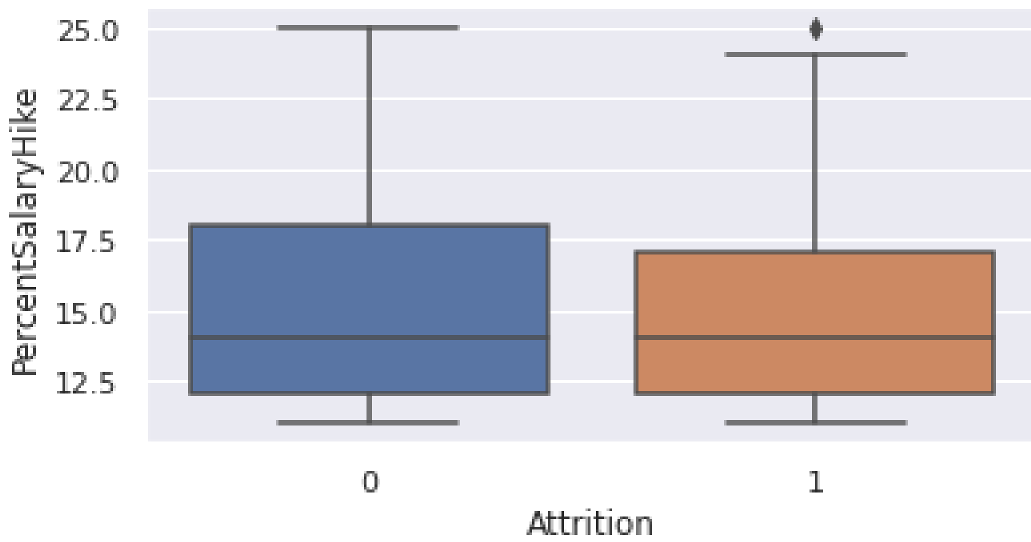
[98]:
```
LeveneResult(statistic=0.3448686107237524, pvalue=0.5571226657108874)
```

[99]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=True)
```

[99]:
```
Ttest_indResult(statistic=-0.5164573250747643, pvalue=0.6056128238893757)
```

[100]:
```
sn.boxplot(x='Attrition', y='PercentSalaryHike', data=data)
```

[100]:
```
<AxesSubplot:xlabel='Attrition', ylabel='PercentSalaryHike'>
```

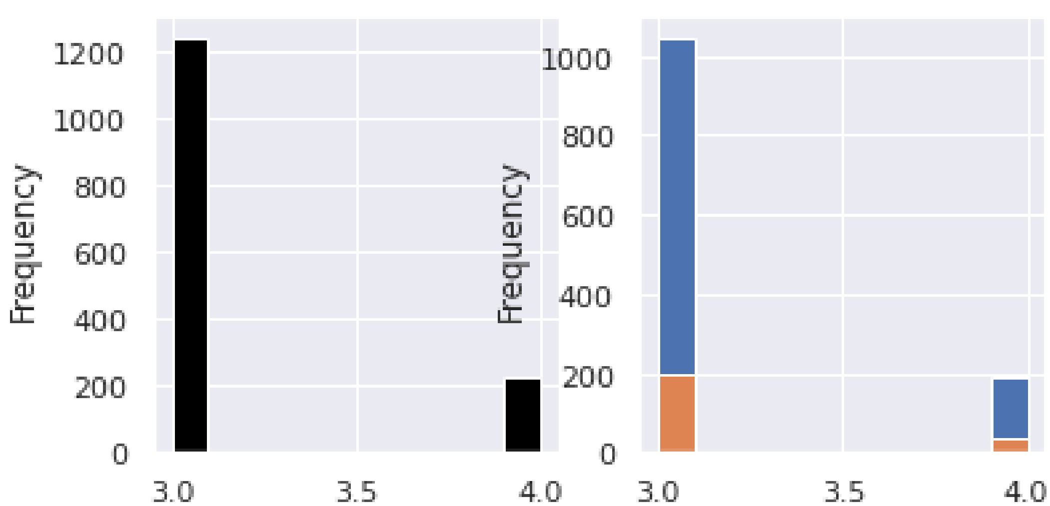


Because we already have a variable that encodes income, and because there is a skewed distribution that might cause issues with our model, we will drop this predictor from our analysis.

PerformanceRating

[101]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['PerformanceRating'].plot(ax=axes[0],kind='hist',color='black')
data['PerformanceRating'].groupby(data['Attrition']).
  ↪plot(ax=axes[1],kind='hist')
```

[101]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: PerformanceRating, dtype: object
```



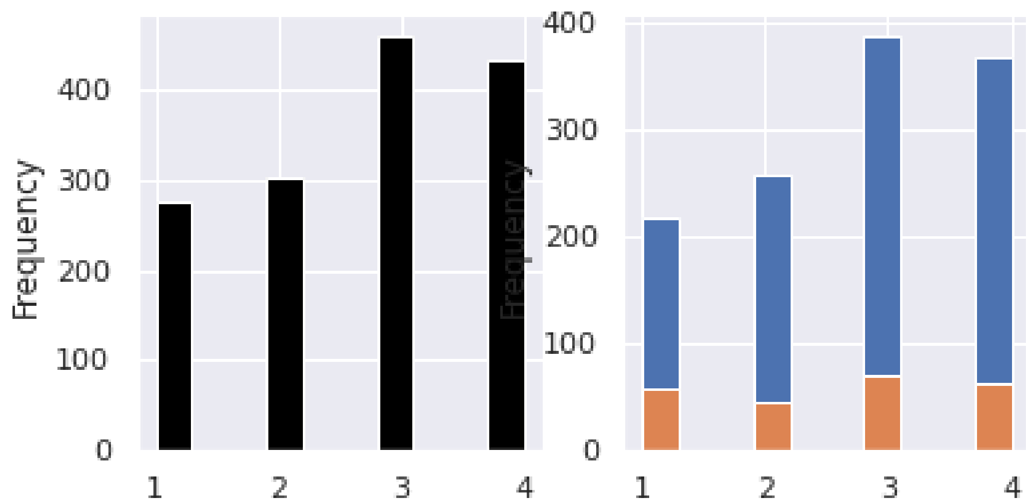
This is another categorical variable. We see that out of the total scale, we only have individuals with a high performance rating. PerformanceRating 1 'Low' 2 'Good' 3 'Excellent' 4 'Outstanding' Because we do not have much variance in this predictor, and because we could interpret this predictor to mean high performers such that all members fall within the same value, we can eliminate this predictor from our analysis.

[102]:
```
data = data.drop(columns=['PerformanceRating'])
```

RelationshipSatisfaction

[103]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['RelationshipSatisfaction'].plot(ax=axes[0],kind='hist',color='black')
data['RelationshipSatisfaction'].groupby(data['Attrition']).
  ↪plot(ax=axes[1],kind='hist')
```

[103]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: RelationshipSatisfaction, dtype: object
```

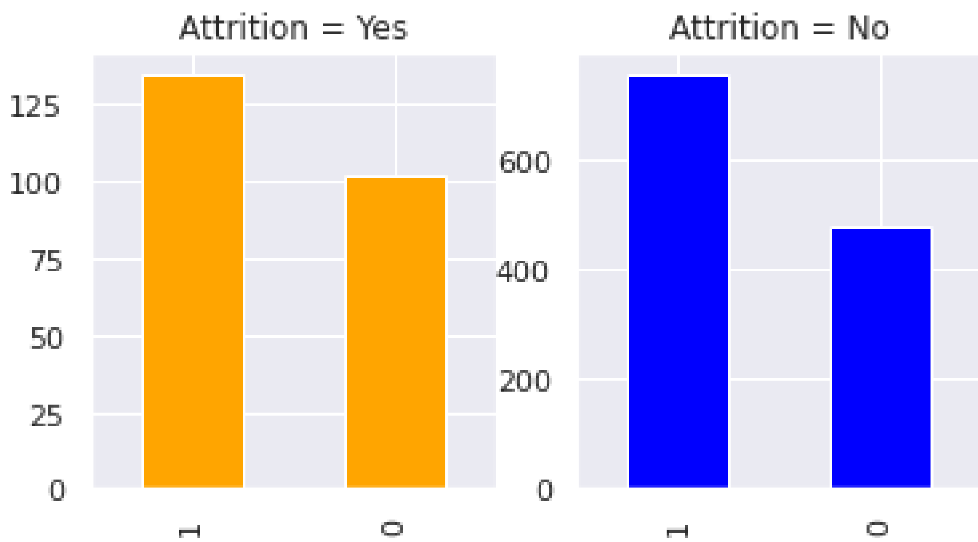


Once again, a categorical variable. RelationshipSatisfaction 1 'Low' 2 'Medium' 3 'High' 4 'Very High' Let's encode it as a boolean variable, to more accurately encode the predictor. 1,2 - less satisfied 3,4 - more satisfied

[104]:
```
data['RelationshipSatisfaction'] = data['RelationshipSatisfaction'].replace(1,0)
data['RelationshipSatisfaction'] = data['RelationshipSatisfaction'].replace(2,0)
data['RelationshipSatisfaction'] = data['RelationshipSatisfaction'].replace(3,1)
data['RelationshipSatisfaction'] = data['RelationshipSatisfaction'].replace(4,1)
```

[105]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 1]
data_attrit['RelationshipSatisfaction'].value_counts().
 ↪plot(kind='bar',title='Attrition = Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 0]
data_stay['RelationshipSatisfaction'].value_counts().
 ↪plot(kind='bar',title='Attrition = No',color='blue',ax=axes[1])
```

[105]:
```
<AxesSubplot:title={'center':'Attrition = No'}>
```



[106]:
```
data['RelationshipSatisfaction'].groupby(data['Attrition']).describe()
```

[106]:

Attrition	count	mean	std	min	25%	50%	75%	max
0	1233.0	0.613139	0.487229	0.0	0.0	1.0	1.0	1.0
1	237.0	0.569620	0.496177	0.0	0.0	1.0	1.0	1.0

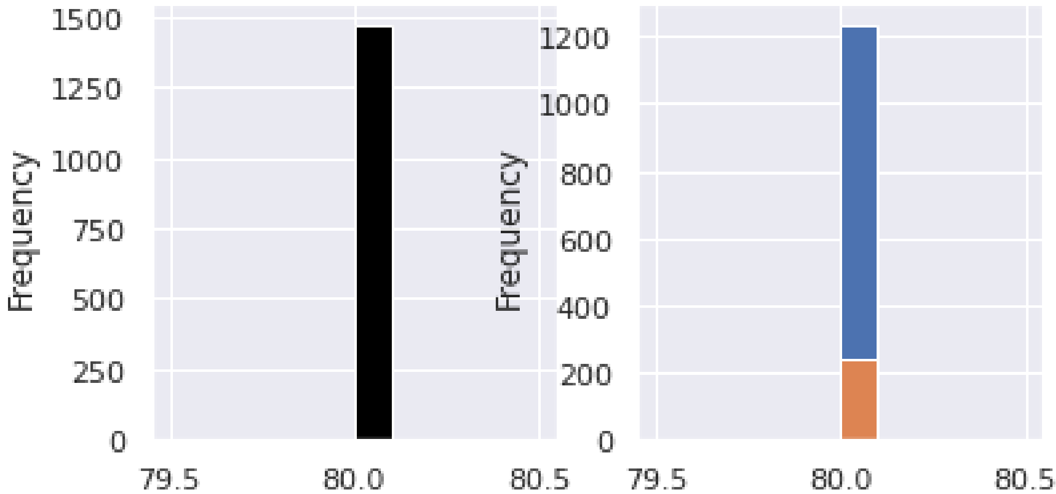
We will keep this variable for now, although we suspect, given our domain knowledge, it might be intercorrelated with other variables like marital status.

StandardHours

[107]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['StandardHours'].plot(ax=axes[0],kind='hist',color='black')
data['StandardHours'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[107]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
```

```
Name: StandardHours, dtype: object
```



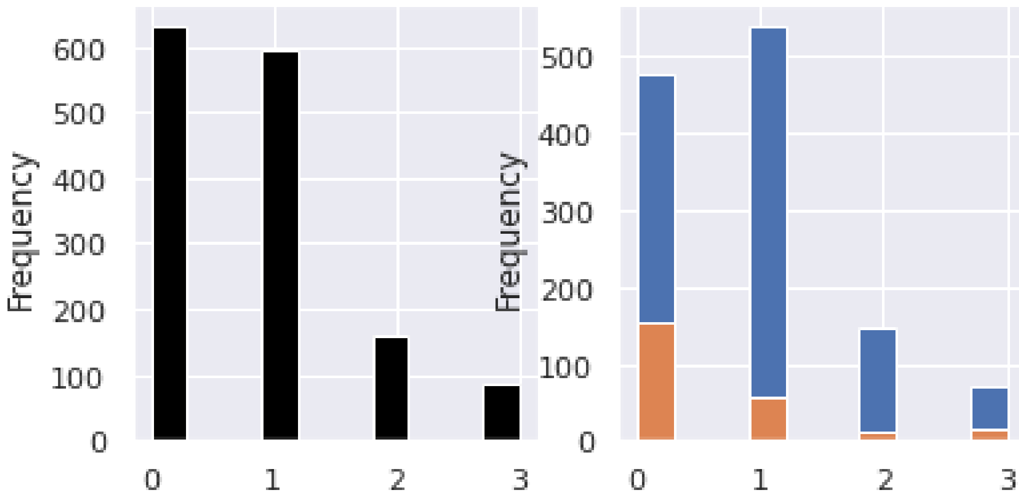
We see that all observations have the same value for this predictor - we can drop this from our analysis.

[108]:
```
data = data.drop(columns=['StandardHours'])
```

Stock Option Level

[109]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['StockOptionLevel'].plot(ax=axes[0],kind='hist',color='black')
data['StockOptionLevel'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[109]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: StockOptionLevel, dtype: object
```



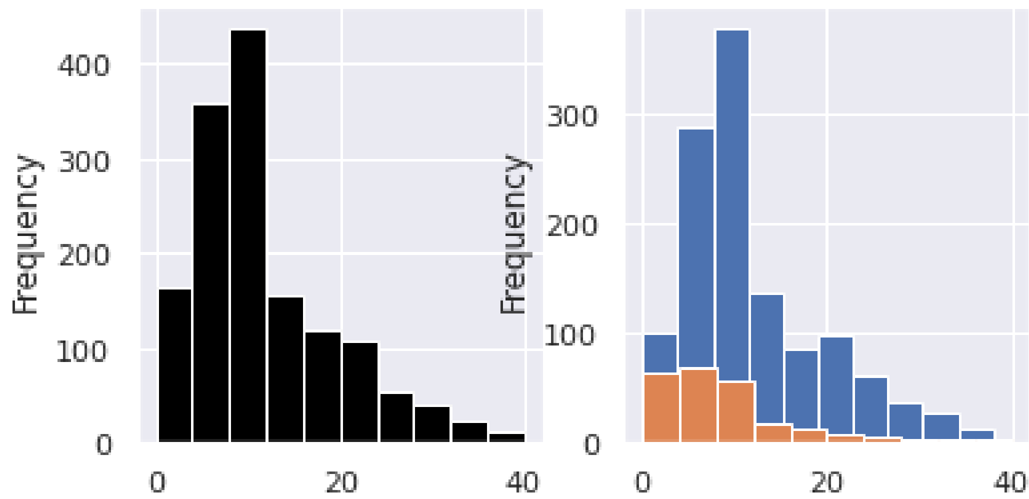
We can see that we have a few levels for this variable that have very small sample sizes - this may cause convergence issues with our model, so we will drop this predictor from our analysis.

[110]:
```
data = data.drop(columns=['StockOptionLevel'])
```

TotalWorkingYears

[111]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['TotalWorkingYears'].plot(ax=axes[0],kind='hist',color='black')
data['TotalWorkingYears'].groupby(data['Attrition']).
 ↪plot(ax=axes[1],kind='hist')
```

[111]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: TotalWorkingYears, dtype: object
```



[112]:
```python
data['TotalWorkingYears'].groupby(data['Attrition']).describe()
```

[112]:

Attrition	count	mean	std	min	25%	50%	75%	max
0	1233.0	11.862936	7.760719	0.0	6.0	10.0	16.0	38.0
1	237.0	8.244726	7.169204	0.0	3.0	7.0	10.0	40.0

[113]:
```python
attrit_rate = data.query('Attrition == 1')['TotalWorkingYears']
stay_rate = data.query('Attrition == 0')['TotalWorkingYears']
stats.levene(attrit_rate, stay_rate)
```

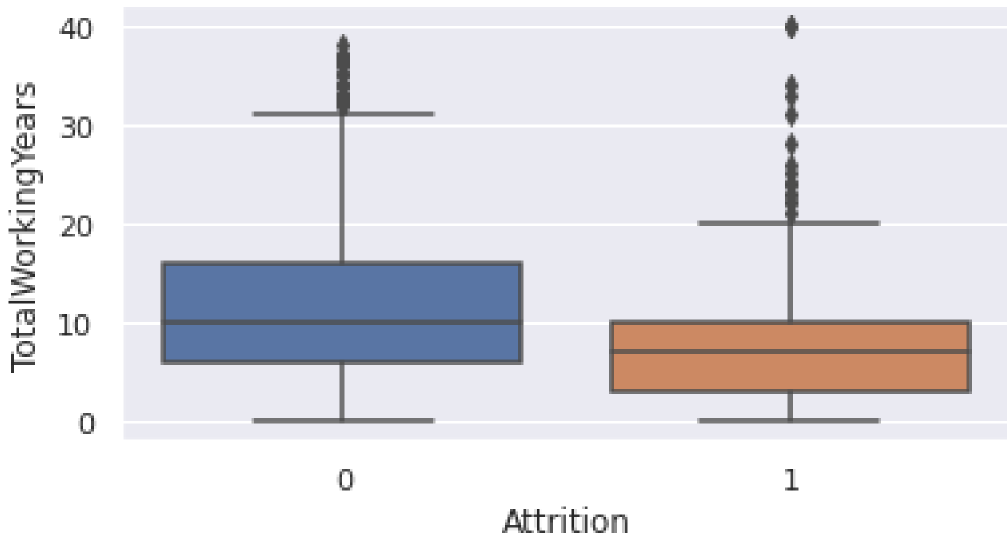
[113]:
```
LeveneResult(statistic=3.0623848089541714, pvalue=0.08033291448975143)
```

[114]:
```python
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=True)
```

[114]:
```
Ttest_indResult(statistic=-6.6522546135024445, pvalue=4.0618781112668525e-11)
```

[115]:
```python
sn.boxplot(x='Attrition', y='TotalWorkingYears', data=data)
```

[115]:
```
<AxesSubplot:xlabel='Attrition', ylabel='TotalWorkingYears'>
```



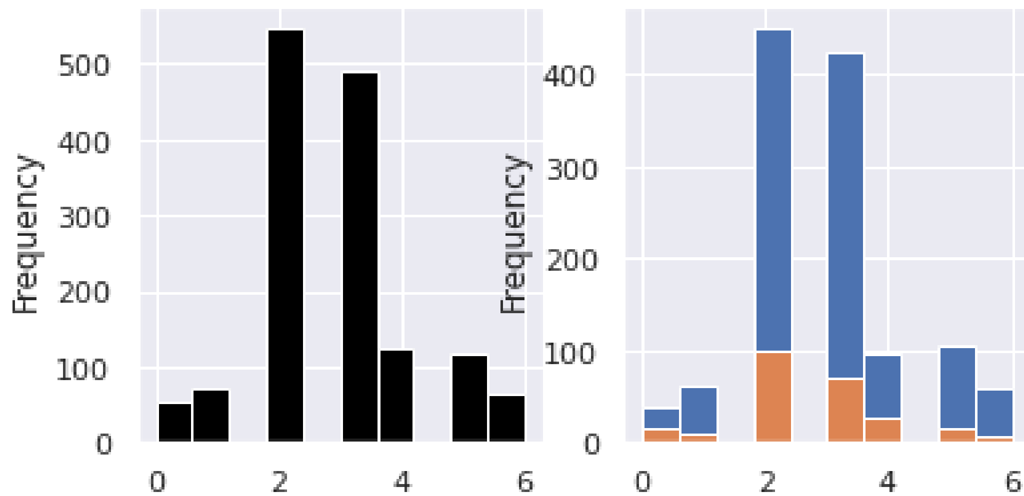
It appears that there are quite a few outliers that might be driving the difference in means - we will drop this predictor from our analysis, as this might cause convergence issues later.

[116]:
```
data = data.drop(columns=['TotalWorkingYears'])
```

TrainingTimesLastYear

[117]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['TrainingTimesLastYear'].plot(ax=axes[0],kind='hist',color='black')
data['TrainingTimesLastYear'].groupby(data['Attrition']).
 ↪plot(ax=axes[1],kind='hist')
```

[117]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: TrainingTimesLastYear, dtype: object
```



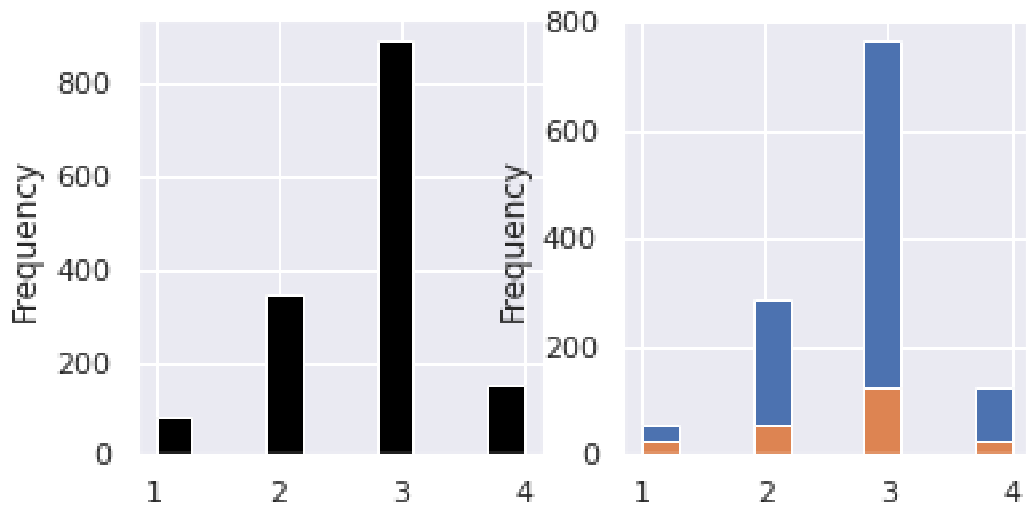
Again, we see that there a few values that have very small sample sizes - because this is a discrete variable, this may impact our analysis, so we will drop this variable from our analysis.

[118]:
```
data = data.drop(columns=['TrainingTimesLastYear'])
```

WorkLifeBalance

[119]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['WorkLifeBalance'].plot(ax=axes[0],kind='hist',color='black')
data['WorkLifeBalance'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[119]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: WorkLifeBalance, dtype: object
```



This is our final categorical variable. WorkLifeBalance 1 'Bad' 2 'Good' 3 'Better' 4 'Best' We can encode this as having less or more worklife balance to better capture the categorical nature of this variable.

[120]:
```
data['WorkLifeBalance'] = data['WorkLifeBalance'].replace(1,0)
data['WorkLifeBalance'] = data['WorkLifeBalance'].replace(2,0)
data['WorkLifeBalance'] = data['WorkLifeBalance'].replace(3,1)
data['WorkLifeBalance'] = data['WorkLifeBalance'].replace(4,1)
```

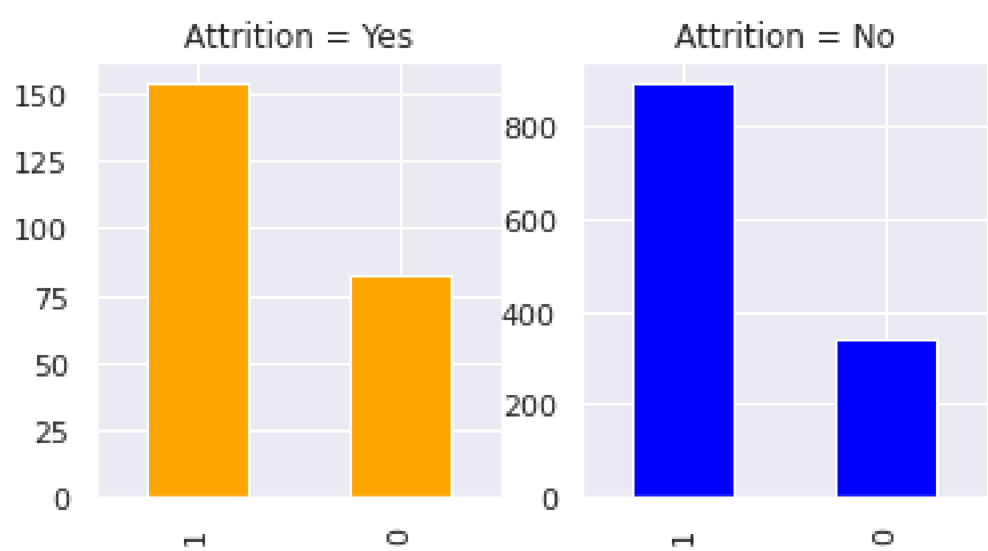
[121]:
```
data['WorkLifeBalance'].groupby(data['Attrition']).describe()
```

[121]:

Attrition	count	mean	std	min	25%	50%	75%	max
0	1233.0	0.723439	0.447479	0.0	0.0	1.0	1.0	1.0
1	237.0	0.649789	0.478045	0.0	0.0	1.0	1.0	1.0

[122]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data_attrit = data.loc[data['Attrition'] == 1]
data_attrit['WorkLifeBalance'].value_counts().plot(kind='bar',title='Attrition␣
↪= Yes',color='orange',ax=axes[0])
data_stay = data.loc[data['Attrition'] == 0]
data_stay['WorkLifeBalance'].value_counts().plot(kind='bar',title='Attrition =␣
↪No',color='blue',ax=axes[1])
```

[122]: <AxesSubplot:title={'center':'Attrition = No'}>

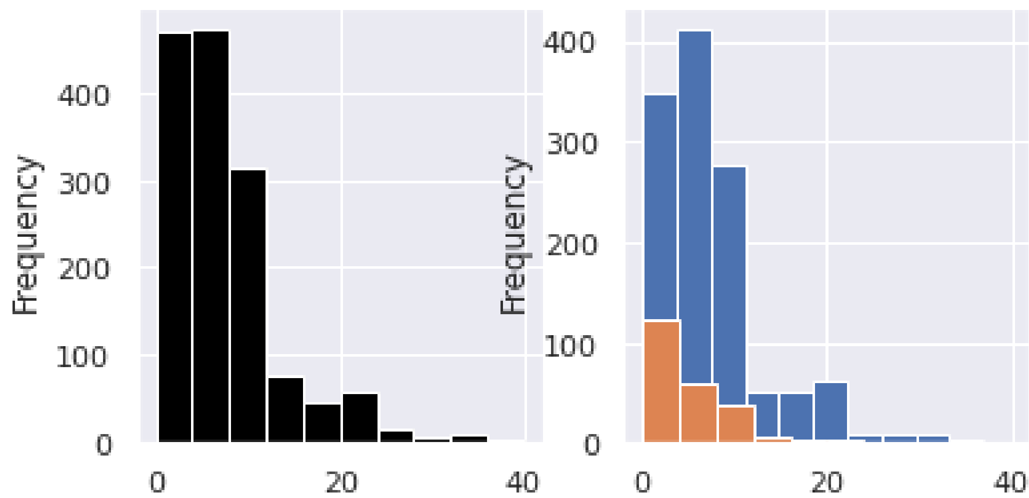


By inspection, we notice that there might be a difference between the two groups.

Years At Company

[123]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['YearsAtCompany'].plot(ax=axes[0],kind='hist',color='black')
data['YearsAtCompany'].groupby(data['Attrition']).plot(ax=axes[1],kind='hist')
```

[123]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: YearsAtCompany, dtype: object
```



[124]:
```
data['YearsAtCompany'].groupby(data['Attrition']).describe()
```

[124]:

	count	mean	std	min	25%	50%	75%	max
Attrition								
0	1233.0	7.369019	6.096298	0.0	3.0	6.0	10.0	37.0
1	237.0	5.130802	5.949984	0.0	1.0	3.0	7.0	40.0

[125]:
```
attrit_rate = data.query('Attrition == 1')['YearsAtCompany']
stay_rate = data.query('Attrition == 0')['YearsAtCompany']
stats.levene(attrit_rate, stay_rate)
```

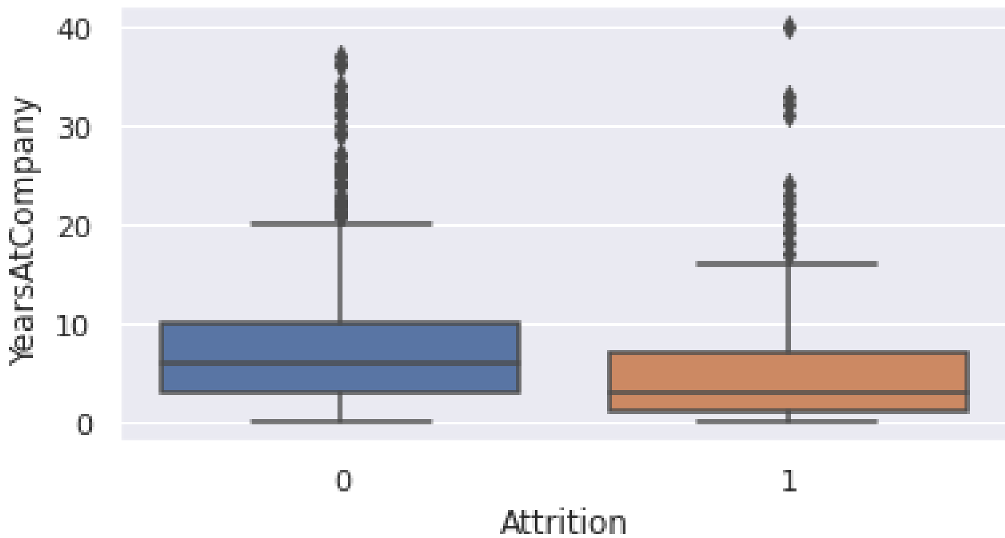
[125]:
```
LeveneResult(statistic=2.7533975962582904, pvalue=0.09726160332756505)
```

[126]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=True)
```

[126]:
```
Ttest_indResult(statistic=-5.1963086670254235, pvalue=2.3188716103863033e-07)
```

[127]:
```
sn.boxplot(x='Attrition', y='YearsAtCompany', data=data)
```

[127]: <AxesSubplot:xlabel='Attrition', ylabel='YearsAtCompany'>

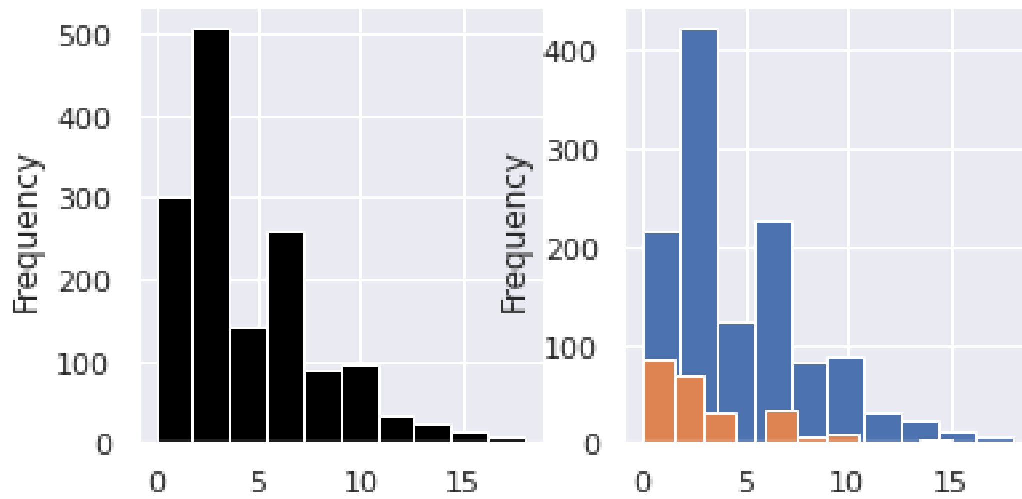


This variable has a very skewed distribution - we may consider dropping this predictor from our analysis as a result.

YearsInCurrentRole

[128]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['YearsInCurrentRole'].plot(ax=axes[0],kind='hist',color='black')
data['YearsInCurrentRole'].groupby(data['Attrition']).
  ↪plot(ax=axes[1],kind='hist')
```

[128]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: YearsInCurrentRole, dtype: object
```



[129]:
```
data['YearsInCurrentRole'].groupby(data['Attrition']).describe()
```

[129]:

Attrition	count	mean	std	min	25%	50%	75%	max
0	1233.0	4.484185	3.649402	0.0	2.0	3.0	7.0	18.0
1	237.0	2.902954	3.174827	0.0	0.0	2.0	4.0	15.0

[130]:
```
attrit_rate = data.query('Attrition == 1')['YearsInCurrentRole']
stay_rate = data.query('Attrition == 0')['YearsInCurrentRole']
stats.levene(attrit_rate, stay_rate)
```

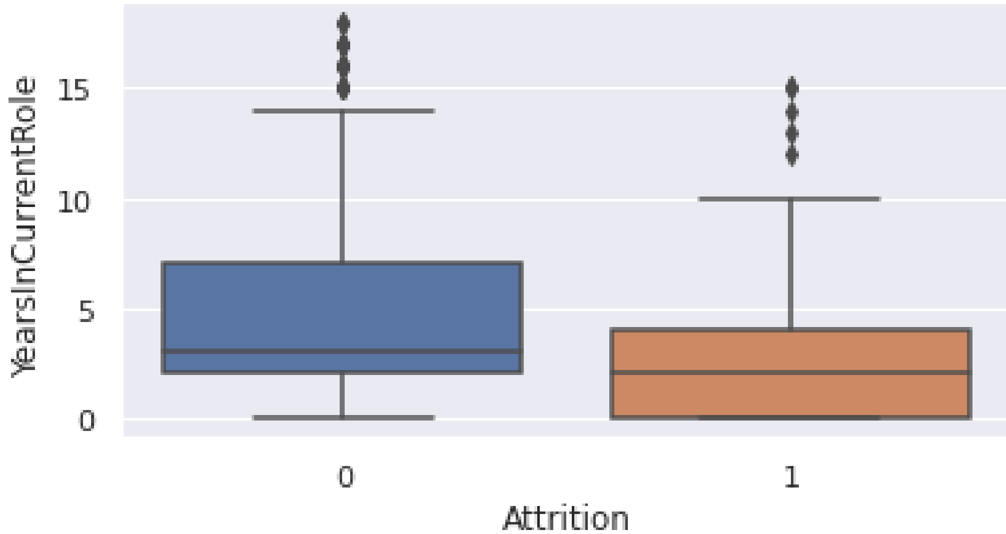
[130]: LeveneResult(statistic=16.023010657349747, pvalue=6.570169050687587e-05)

[131]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=False)
```

[131]: Ttest_indResult(statistic=-6.847079159882748, pvalue=3.1873903722051294e-11)

[132]:
```
sn.boxplot(x='Attrition', y='YearsInCurrentRole', data=data)
```

[132]: <AxesSubplot:xlabel='Attrition', ylabel='YearsInCurrentRole'>

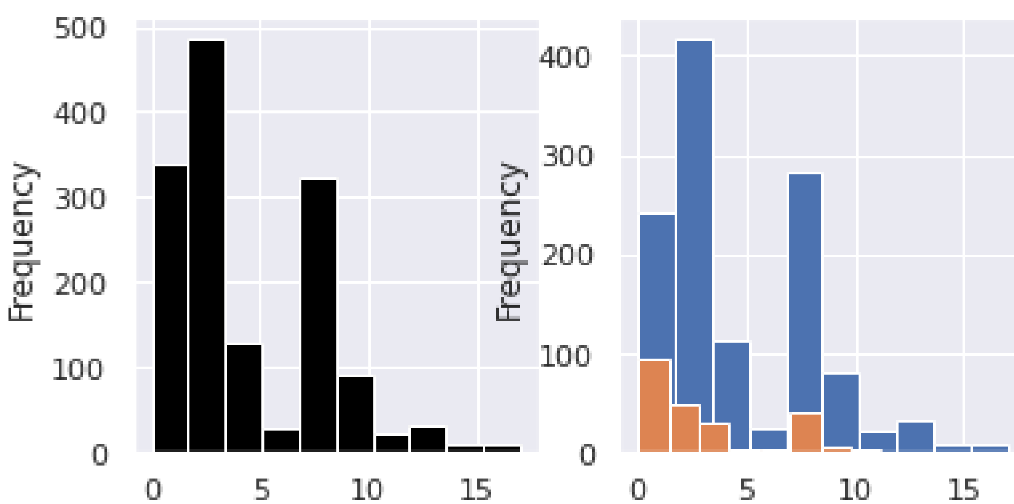


This is also a highly skewed distribution, with a good number of outliers.

YearsWithCurrManager

[133]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['YearsWithCurrManager'].plot(ax=axes[0],kind='hist',color='black')
data['YearsWithCurrManager'].groupby(data['Attrition']).
  ↪plot(ax=axes[1],kind='hist')
```

[133]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: YearsWithCurrManager, dtype: object
```



[134]:
```
data['YearsWithCurrManager'].groupby(data['Attrition']).describe()
```

[134]:
```
            count      mean       std  min  25%  50%  75%   max
Attrition
0          1233.0  4.367397  3.594116  0.0  2.0  3.0  7.0  17.0
1           237.0  2.852321  3.143349  0.0  0.0  2.0  5.0  14.0
```

[135]:
```
attrit_rate = data.query('Attrition == 1')['YearsWithCurrManager']
stay_rate = data.query('Attrition == 0')['YearsWithCurrManager']
stats.levene(attrit_rate, stay_rate)
```

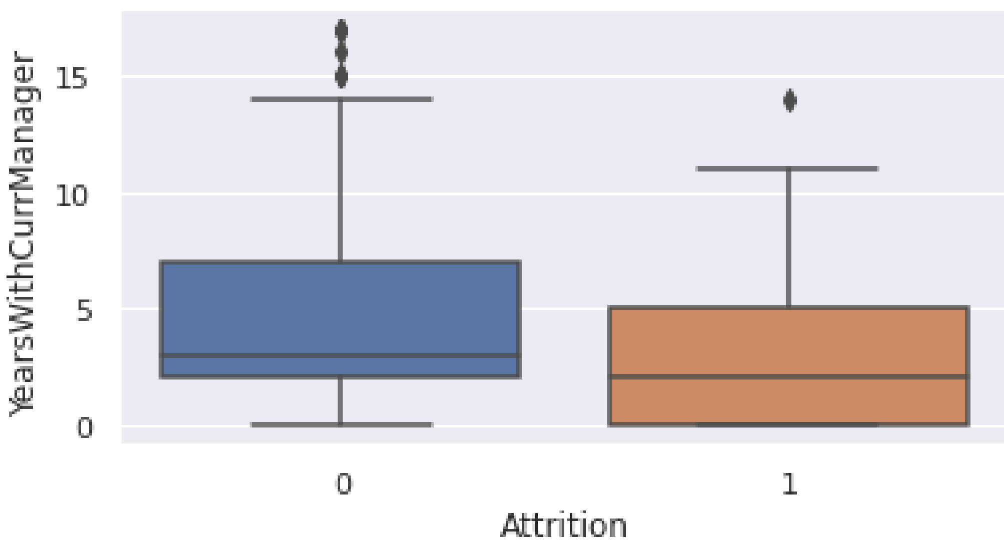
[135]:
```
LeveneResult(statistic=9.66176551705255, pvalue=0.0019175509895717584)
```

[136]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=False)
```

[136]:
```
Ttest_indResult(statistic=-6.6333988161585, pvalue=1.1850219000030649e-10)
```

[137]:
```
sn.boxplot(x='Attrition', y='YearsWithCurrManager', data=data)
```

[137]:
```
<AxesSubplot:xlabel='Attrition', ylabel='YearsWithCurrManager'>
```

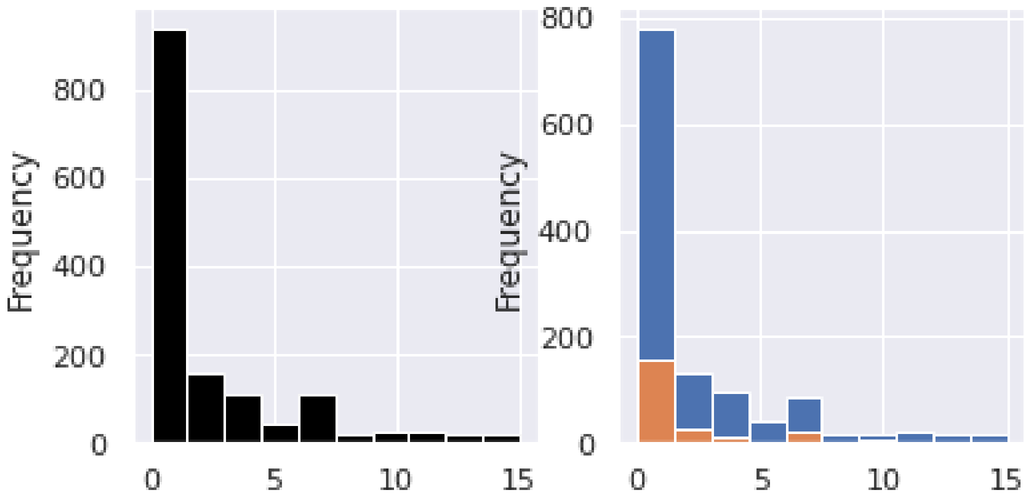


There seems to be a difference between groups, but there are also a number of outliers. We will need to consider this as we continue selecting features.

YearsSinceLastPromotion

[138]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
data['YearsSinceLastPromotion'].plot(ax=axes[0],kind='hist',color='black')
data['YearsSinceLastPromotion'].groupby(data['Attrition']).
 ↪plot(ax=axes[1],kind='hist')
```

[138]:
```
Attrition
0    AxesSubplot(0.547727,0.125;0.352273x0.755)
1    AxesSubplot(0.547727,0.125;0.352273x0.755)
Name: YearsSinceLastPromotion, dtype: object
```



[139]:
```
data['YearsSinceLastPromotion'].groupby(data['Attrition']).describe()
```

[139]:

Attrition	count	mean	std	min	25%	50%	75%	max
0	1233.0	2.234388	3.234762	0.0	0.0	1.0	3.0	15.0
1	237.0	1.945148	3.153077	0.0	0.0	1.0	2.0	15.0

[140]:
```
attrit_rate = data.query('Attrition == 1')['YearsSinceLastPromotion']
stay_rate = data.query('Attrition == 0')['YearsSinceLastPromotion']
stats.levene(attrit_rate, stay_rate)
```

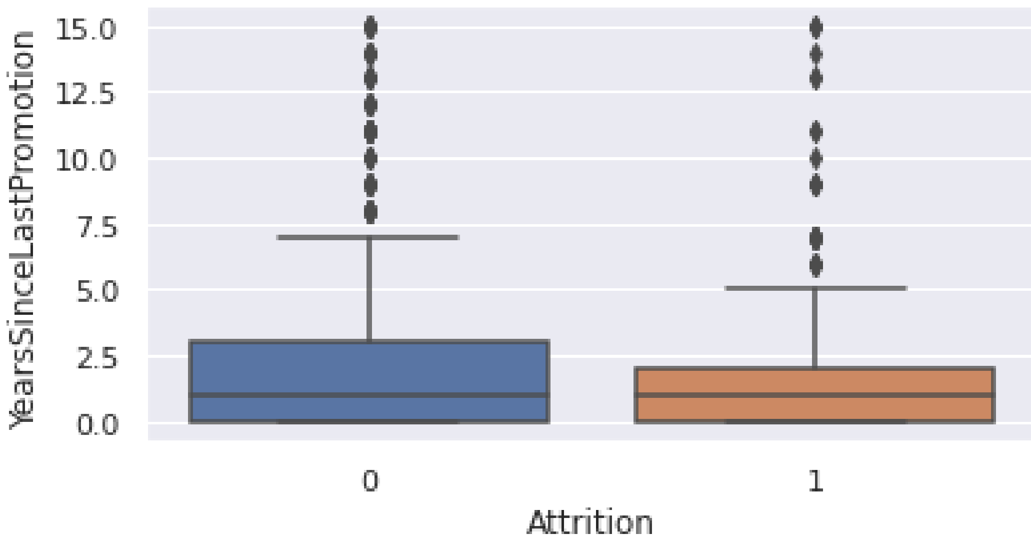
[140]:
```
LeveneResult(statistic=0.39377468332250853, pvalue=0.5304195027928351)
```

[141]:
```
import scipy.stats as stats
stats.ttest_ind(attrit_rate,stay_rate,equal_var=True)
```

[141]:
```
Ttest_indResult(statistic=-1.2657876620135298, pvalue=0.2057899591624936)
```

[142]:
```
sn.boxplot(x='Attrition', y='YearsSinceLastPromotion', data=data)
```

[142]: <AxesSubplot:xlabel='Attrition', ylabel='YearsSinceLastPromotion'>



This is an incredibly skewed distribution, with many outliers. We will drop this predictor from our dataset.

[143]:
```
data = data.drop(columns=['YearsSinceLastPromotion'])
```

[144]:
```
data2 = data
```

Summary We have explored the distributions of our continuous numerical variables, and done some data cleaning and investigation, based on available domain knowledge. We notice that very few of our predictors likely have a normal distribution within each class, so therefore, we can rule out classification methods like linear discriminant analysis. We also notice that there is not obvious separation between the classes within each predictor - this suggests that logistic regression is a stable approach.

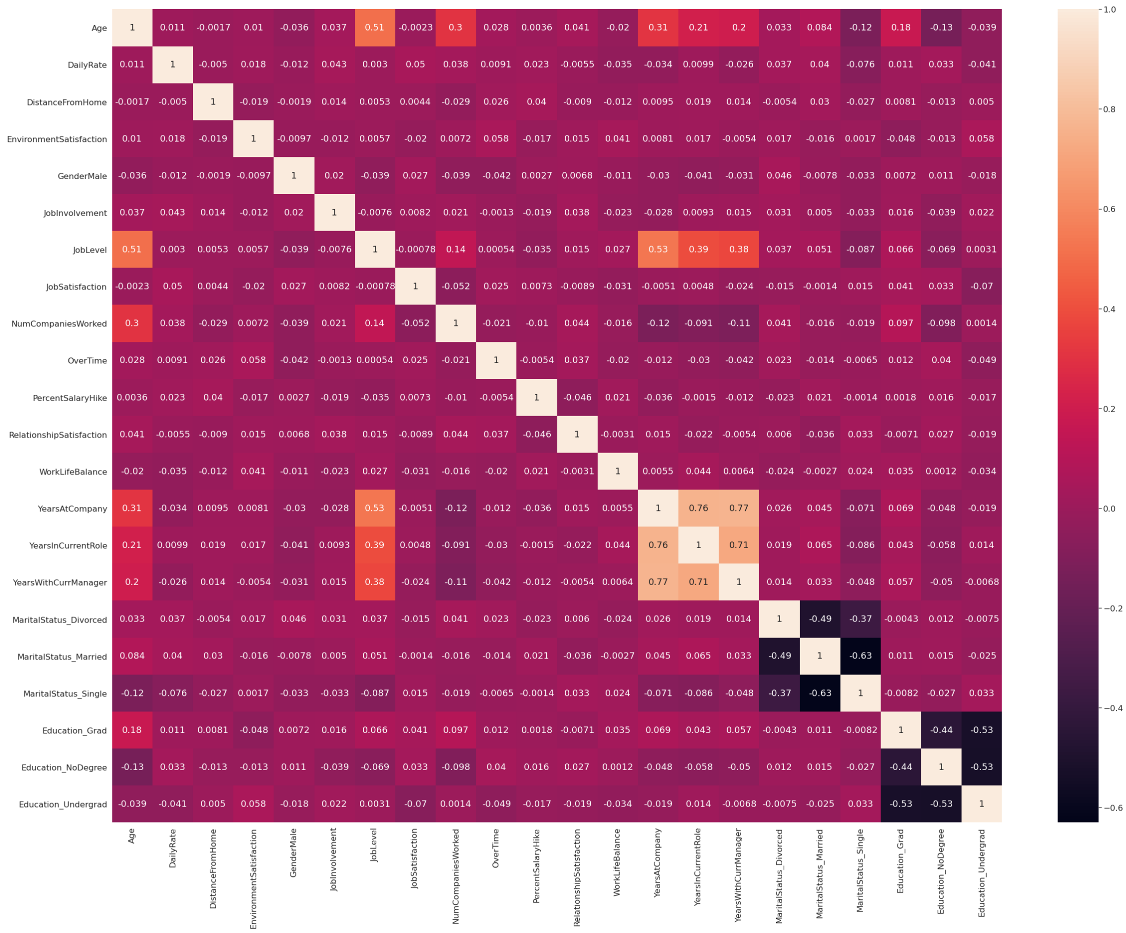
Now that we have prepared all of our variables, we need to investigate collinearity, and remove strongly correlated predictors.

Collinearity Predictors that are highly collinear need to be dropped from our analysis. We can either combine predictors, create new predictors, or drop highly correlated factors from our analysis.

[145]:
```
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Our first approach will be to leverage a heatmap, so we can see highly correlated features easily.

```
[146]: data_corr = data.drop(['Attrition'],axis=1)
corrMatrix = data_corr.corr()
sn.set(rc={'figure.figsize':(40,30)})
sn.set(font_scale=1.5)
sn.heatmap(corrMatrix, annot=True)
plt.show()
```



The cut-off for 'strongly correlated' is usually a score of 0.70 or above. We notice that there are a few moderately correlated features (0.50 or above), but we will not remove them from our predictor set now. Let's remove our strongly correlated features from our predictor set.

```
[147]: data = data.
    ↪drop(columns=['YearsAtCompany','YearsInCurrentRole','YearsWithCurrManager'])
```

Another approach to reducing collinearity and unwanted variance is to use a variance inflation factor calculation, or VIF score, to evaluate factors that contribute the most to model variance.

```
[148]: data_corr = data.drop(['Attrition'],axis=1)
vif = pd.DataFrame()
```

```
vif["features"] = data_corr.columns
vif["vif_Factor"] = [variance_inflation_factor(data_corr.values, i) for i in␣
 ↪range(data_corr.shape[1])]
vif
```

[148]:

	features	vif_Factor
0	Age	1.511601
1	DailyRate	1.015848
2	DistanceFromHome	1.005338
3	EnvironmentSatisfaction	1.011806
4	GenderMale	1.009569
5	JobInvolvement	1.009735
6	JobLevel	1.361918
7	JobSatisfaction	1.013570
8	NumCompaniesWorked	1.115367
9	OverTime	1.013752
10	PercentSalaryHike	1.008354
11	RelationshipSatisfaction	1.011094
12	WorkLifeBalance	1.011037
13	MaritalStatus_Divorced	inf
14	MaritalStatus_Married	inf
15	MaritalStatus_Single	inf
16	Education_Grad	inf
17	Education_NoDegree	inf
18	Education_Undergrad	inf

We will eliminate variables with a VIF score greater than 10.

VIF scores of INF indicate perfect collinearity - we will need to eliminate further variables in order to resolve this issue.

[149]:

```
data = data.
 ↪drop(columns=['Education_Grad','Education_NoDegree','Education_Undergrad','MaritalStatus_Di
```

[150]:

```
data_corr = data.drop(['Attrition'],axis=1)
vif = pd.DataFrame()
vif["features"] = data_corr.columns
vif["vif_Factor"] = [variance_inflation_factor(data_corr.values, i) for i in␣
 ↪range(data_corr.shape[1])]
vif
```

[150]:

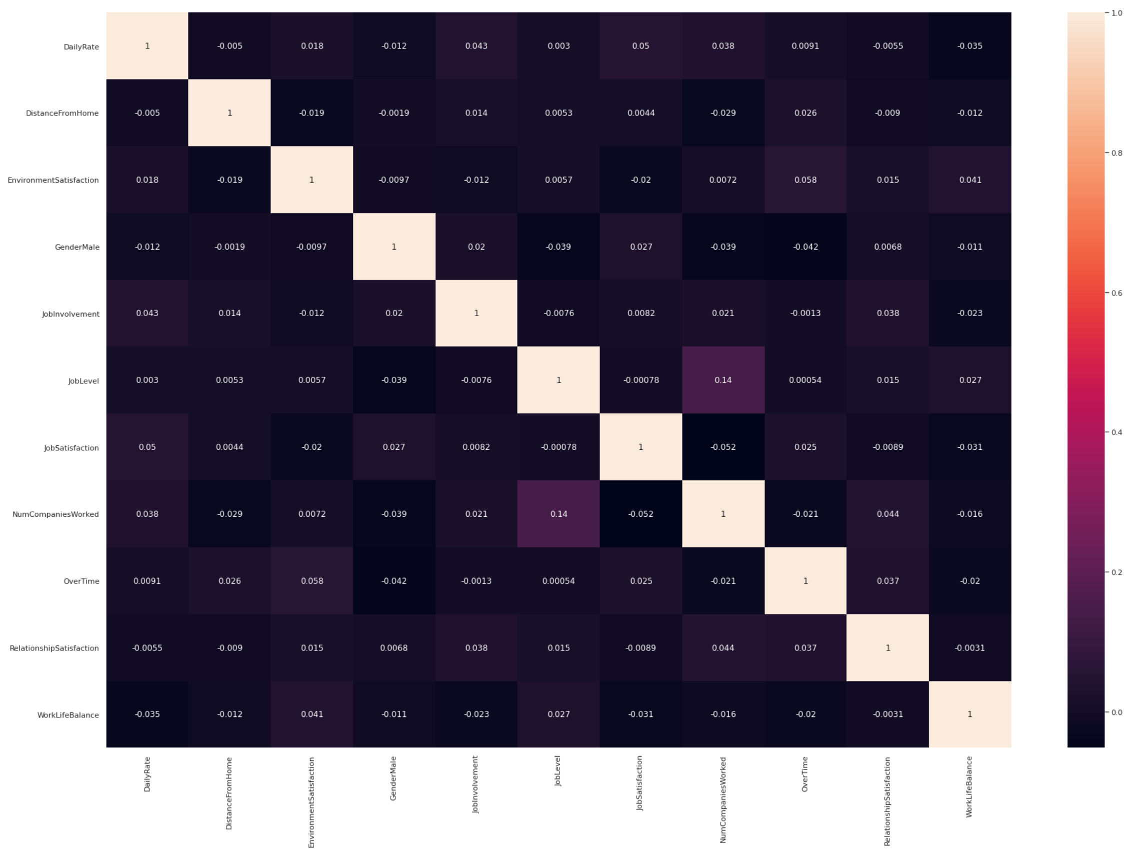
	features	vif_Factor
0	Age	19.783245
1	DailyRate	4.676489
2	DistanceFromHome	2.249682
3	EnvironmentSatisfaction	2.517029
4	GenderMale	2.422625
5	JobInvolvement	3.124185

```
6                      JobLevel    6.085303
7               JobSatisfaction    2.534030
8            NumCompaniesWorked    2.397062
9                      OverTime    1.403532
10           PercentSalaryHike    12.221724
11     RelationshipSatisfaction    2.484607
12              WorkLifeBalance    3.297188
```

As we've removed variables, we can see that there's additional variables that have poor VIF scores now. Let's eliminate those.

[151]:
```
data = data.drop(columns=['Age','PercentSalaryHike'])
```

[152]:
```
data_corr = data.drop(['Attrition'],axis=1)
corrMatrix = data_corr.corr()
sn.set(rc={'figure.figsize':(30,20)})
sn.set(font_scale=1)
sn.heatmap(corrMatrix, annot=True)
plt.show()
```



We can see clearly now that we have reduced collinearity significantly within our predictors.

[153]:
```
data_corr = data.drop(['Attrition'],axis=1)
vif = pd.DataFrame()
vif["features"] = data_corr.columns
vif["vif_Factor"] = [variance_inflation_factor(data_corr.values, i) for i in␣
↪range(data_corr.shape[1])]
vif
```

[153]:

	features	vif_Factor
0	DailyRate	4.161205
1	DistanceFromHome	2.143422
2	EnvironmentSatisfaction	2.427644
3	GenderMale	2.305405
4	JobInvolvement	2.947651
5	JobLevel	3.956010
6	JobSatisfaction	2.420726
7	NumCompaniesWorked	2.153264
8	OverTime	1.387235
9	RelationshipSatisfaction	2.403458
10	WorkLifeBalance	3.033991

Now that we've eliminated variables that contribute to collinearity, we are ready to start selecting variables and fitting a model.

Feature Selection & Modeling

Logistic Regression Model

Principle Component Analysis is an approach that can help us reduce dimensionality, and help us understand what proportion of the variance we are capturing.

Let's start by creating a logistic regression model, to see whether we're able to create a working model given the predictors we curently have, or if there are only a few significant factors we should include in our final model.

[154]:
```
y = data['Attrition']
x = data.drop('Attrition',axis=1)
print('Available Features',x.columns)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.20,␣
↪shuffle=True, random_state=2)
```

```
Available Features Index(['DailyRate', 'DistanceFromHome',
'EnvironmentSatisfaction',
       'GenderMale', 'JobInvolvement', 'JobLevel', 'JobSatisfaction',
       'NumCompaniesWorked', 'OverTime', 'RelationshipSatisfaction',
       'WorkLifeBalance'],
      dtype='object')
```

[155]:
```python
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(max_iter = 2500)
clf.fit(x_train,y_train)
clf.fit(x_train,y_train)
print('Coefficients and Intercept Value',clf.coef_,clf.intercept_)
y_pred = clf.predict(x_test)
y_true = y_test
```

```
Coefficients and Intercept Value [[-4.86883193e-04  3.12415667e-02
-6.59671371e-01  3.47610215e-01
  -6.84516221e-01 -5.21127560e-01 -5.15016658e-01  8.49435641e-02
   1.50266044e+00 -1.86046123e-01 -4.18659682e-01]] [-0.11419202]
```

These are the coefficients and intercept for our model. Unfortunately, we are not able to retrieve any information from our model about the p-values or statistical significance of any of our coefficients or the intercept. However, we can use this information to discuss the relevance of some of our features to the outcome. We might use the coefficients to understand how a one unit change in the predictor impacts the log-likelihood. For example, our first predictor is DailyRate. We can say that in this model, a one unit change in DailyRate corresponds to a decrease in the log likelihood of attrition by 0.0004. Is this enough for us to come to our business leaders and share the relationship between these variables and our outcome? Unfortunately no, as we do not know the statistical significance of any of these features, and whether there are additional interaction effects that we'd like to consider or other contraints.

In order to get some information about our coefficients and features, let's use a statsmodels implementation of the logistic regression model, to see if we can get any additional information.

[156]:
```python
from statsmodels.discrete.discrete_model import Logit
from statsmodels.tools import add_constant
x_train2 = add_constant(x_train)
print(Logit(y_train, x_train2).fit().summary())
```

```
Optimization terminated successfully.
         Current function value: 0.368265
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                 1176
Model:                          Logit   Df Residuals:                     1164
Method:                           MLE   Df Model:                           11
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.1647
Time:                        21:05:32   Log-Likelihood:                -433.08
converged:                       True   LL-Null:                       -518.44
Covariance Type:            nonrobust   LLR p-value:                 8.362e-31
================================================================================
============
                               coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
------------
const                        0.0219      0.387      0.057      0.955      -0.737
0.781
DailyRate                   -0.0005      0.000     -2.404      0.016      -0.001
-9.5e-05
DistanceFromHome             0.0309      0.010      2.963      0.003       0.010
0.051
EnvironmentSatisfaction     -0.6919      0.175     -3.958      0.000      -1.035
-0.349
GenderMale                   0.3414      0.184      1.854      0.064      -0.020
0.702
JobInvolvement              -0.7220      0.180     -4.008      0.000      -1.075
-0.369
JobLevel                    -0.5404      0.099     -5.469      0.000      -0.734
-0.347
JobSatisfaction             -0.5476      0.175     -3.128      0.002      -0.891
-0.205
NumCompaniesWorked           0.0845      0.034      2.471      0.013       0.017
0.152
OverTime                     1.5539      0.178      8.719      0.000       1.205
1.903
RelationshipSatisfaction    -0.1984      0.178     -1.113      0.266      -0.548
0.151
WorkLifeBalance             -0.4576      0.183     -2.503      0.012      -0.816
-0.099
================================================================================
============
```

We can see here that our model does not take into account interaction effects, and does not match the model created by our other method. This is to be expected, as we're using different methods. If we investigate the p-values, we notice that there are a few significant factors. DailyRate,DistanceFromHome,EnvironmentSatisfaction,JobInvolvement,JobLevel,JobSatisfaction,NumCompaniesWo and OverTime are significant. Let's rebuild the model, only including these factors.

[157]:
```python
x_train2 = add_constant(x_train)
x_train2 = x_train2.
 ↪drop(columns=['GenderMale','RelationshipSatisfaction','WorkLifeBalance'])
model = Logit(y_train, x_train2).fit()
print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.372889
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                 1176
Model:                          Logit   Df Residuals:                     1167
Method:                           MLE   Df Model:                            8
```

```
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.1542
Time:                        21:05:35   Log-Likelihood:                -438.52
converged:                       True   LL-Null:                       -518.44
Covariance Type:            nonrobust   LLR p-value:                 1.719e-30
================================================================================
===========
                              coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
const                      -0.2038      0.327     -0.623      0.533      -0.845
0.437
DailyRate                  -0.0005      0.000     -2.209      0.027      -0.001
-5.27e-05
DistanceFromHome            0.0316      0.010      3.054      0.002       0.011
0.052
EnvironmentSatisfaction    -0.7141      0.174     -4.114      0.000      -1.054
-0.374
JobInvolvement             -0.6997      0.179     -3.919      0.000      -1.050
-0.350
JobLevel                   -0.5554      0.099     -5.619      0.000      -0.749
-0.362
JobSatisfaction            -0.5137      0.174     -2.959      0.003      -0.854
-0.173
NumCompaniesWorked          0.0811      0.034      2.389      0.017       0.015
0.148
OverTime                    1.5295      0.176      8.684      0.000       1.184
1.875
================================================================================
===========
```

We see that our coefficients remained significant, but no improvement in our intercept. The confidence interval contains 0, which tells us that when x = 0, the log odds of having attrition as an outcome are likely 0. The two models are performing similarly, which means that perhaps we can drop the other variables from our analysis.

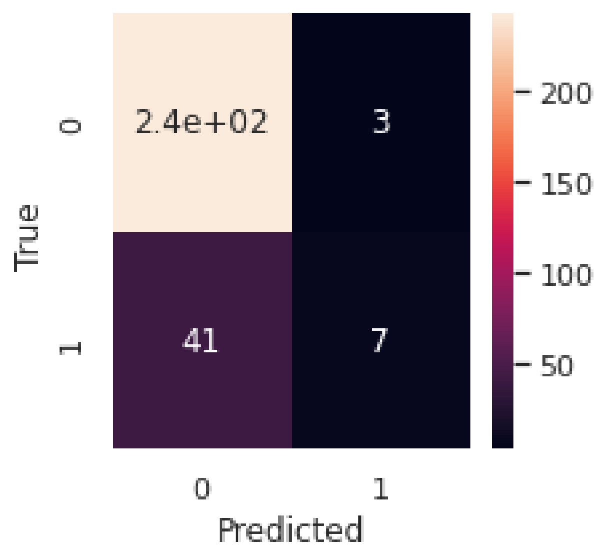
However, it could also mean that there are interaction effects we are missing. Our sklearn model does more to incorporate polynomial and interaction effects. Let's see how our sklearn model performed, so we know if we're able to build a logistic regression model that performs sufficiently well. If we are not able to build a logistic regression model that performs sufficiently well, then perhaps we are using the wrong model, which might suggest that the decision boundary is very non-linear and flexible.

[158]:
```
import numpy as np
from sklearn.metrics import accuracy_score
print("Training accuracy:")
print(np.round(accuracy_score(y_train,clf.predict(x_train)),2))
print("Test accuracy:")
print(np.round(accuracy_score(y_true,y_pred),2))
```

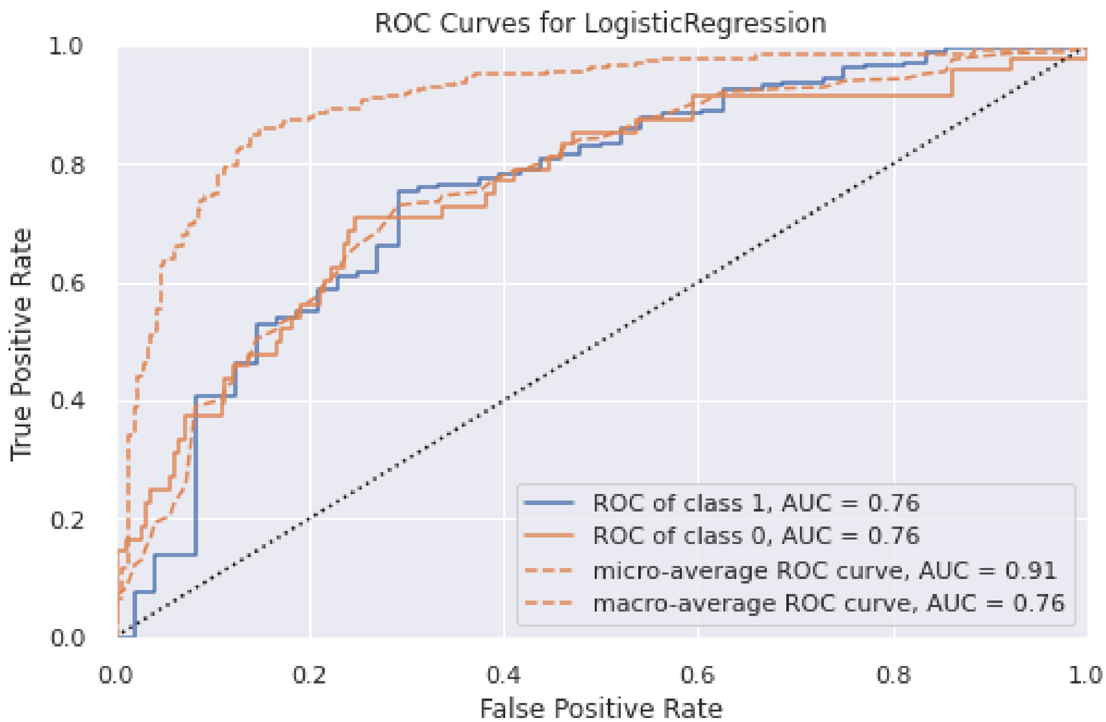
```
from sklearn.metrics import confusion_matrix
sn.set(rc={'figure.figsize':(3,3)})
sn.set(font_scale=1)
matrix = confusion_matrix(y_true,y_pred)
sn.heatmap(matrix,annot=True)
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
Training accuracy:
0.86
Test accuracy:
0.85
```

[158]:
```
Text(3.5, 0.5, 'True')
```



[159]:
```
from yellowbrick.classifier import ROCAUC
sn.set(rc={'figure.figsize':(8,5)})
visualizer = ROCAUC(clf,classes=[1,0])
visualizer.fit(x_train.values, y_train)
visualizer.score(x_test, y_test)
visualizer.show()
```



[159]: <AxesSubplot:title={'center':'ROC Curves for LogisticRegression'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>

When looking at our ROC curve, we want to investigate our micro-average ROC value, since we have unbalanced class sizes. This gives us a great ROC score for our model.

So we're clearly able to create a (relatively) accurate model using a logistic regression approach that incorporates all of our current predictors. We can see that we have a relatively low false positive rate, but a relatively higher false negative rate. Let's see if principle component analysis can help us reduce the dimensionality, and potentially improve our model's performance.

Principal Component Analysis

[160]:
```
from sklearn.preprocessing import StandardScaler

y = data['Attrition']
x = data.drop('Attrition',axis=1)

scaler = StandardScaler()
x = scaler.fit_transform(x)
```

[161]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components = None)
pca.fit(x)
```

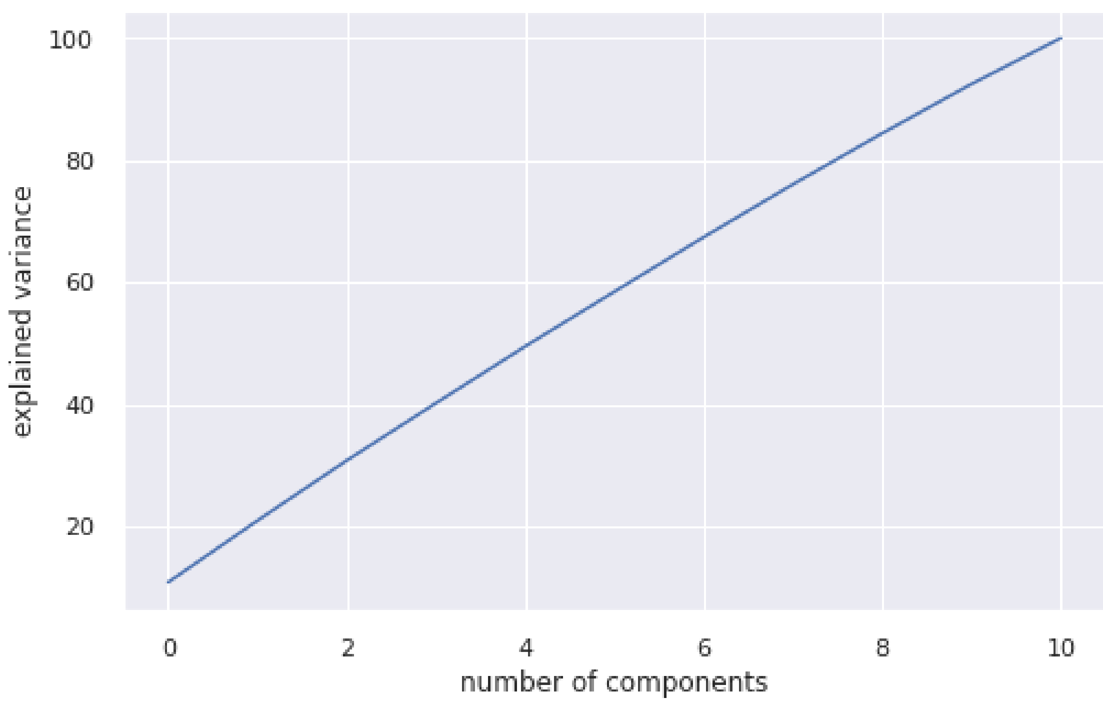
[161]: PCA()

[162]:
```
print('Variance Explained - %')
print(pca.explained_variance_ratio_ * 100)
```

```
Variance Explained - %
[10.81905112 10.11529244  9.84157375  9.41441669  9.2943261   8.97426177
  8.93572742  8.66026532  8.35614929  8.0661466   7.52278951]
```

[163]:
```
plt.plot(np.cumsum(pca.explained_variance_ratio_*100))
plt.xlabel('number of components')
plt.ylabel('explained variance')
```

[163]: Text(0, 0.5, 'explained variance')



We can see that we likely need all of our included predictors in order to explain enough of the variance. Unfortunately, with this approach, we can't retrieve p-values or other statistical measures to identify significance of any of these features. What this approach can tell us is whether dimension reduction should be explored (meaning that there are unneeded variables that we can drop from our analysis), as well as whether a logistic regression model (or other linear model) can be used to model our data. Because we are getting relatively good accuracy with our model, we have sufficient evidence that we can build a relatively good classifier just with a logistic regression model. Additionally, because each of our features seems to explain a relatively equal amount of variance (which is visualized by the linear plot above), we would not want to explore dimension reduction. This makes any simple logistic regression model difficult to fit manually, particularly with interaction effects, because we have so many features. It also rules out some non-parametric methods like K-Nearest Neighbors, which suffer from the curse of dimensionality. This is not necessarily surprising, given that we're working with a simulated dataset. Unfortunately, this means we are constrained to fitting models with most (if not all) of our current predictors, with none of them particularly more important than another.

We can see that we need all of our predictors that we curently have in order to explain a large proportion of the variance. For example, we could look at our original dataset, and see the difference in the shape of the curve.

[164]:
```
y = data2['Attrition']
x = data2.drop('Attrition',axis=1)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.20,␣
 ↪shuffle=True, random_state=2)
```

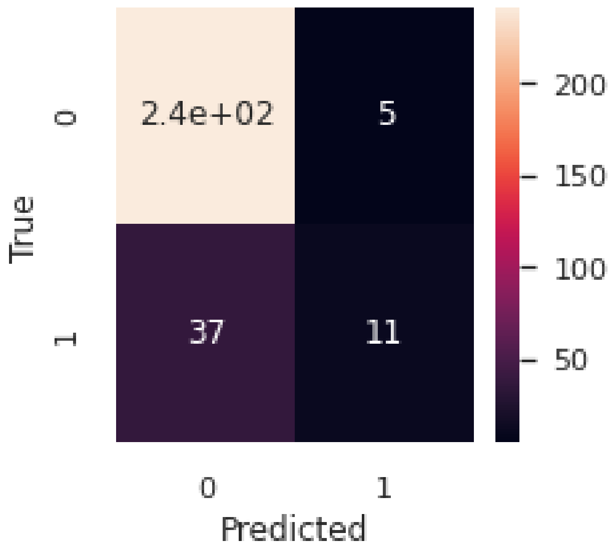
[165]:
```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(max_iter = 2500)
clf.fit(x_train,y_train)
print(clf.coef_,clf.intercept_)
y_pred = clf.predict(x_test)
y_true = y_test
```

```
[[-4.01931431e-02 -3.87113694e-04  3.69115417e-02 -6.91870722e-01
   3.90152196e-01 -6.12167694e-01 -3.46976445e-01 -6.28506388e-01
   1.19136262e-01  1.62643921e+00 -1.59687368e-02 -2.83262485e-01
  -5.17870791e-01  7.94042672e-02 -1.00768865e-01 -9.67570081e-02
  -3.60854674e-01  1.03640760e-02  9.91690466e-01  2.83111536e-01
   1.27603698e-01  2.30484634e-01]] [0.74751367]
```

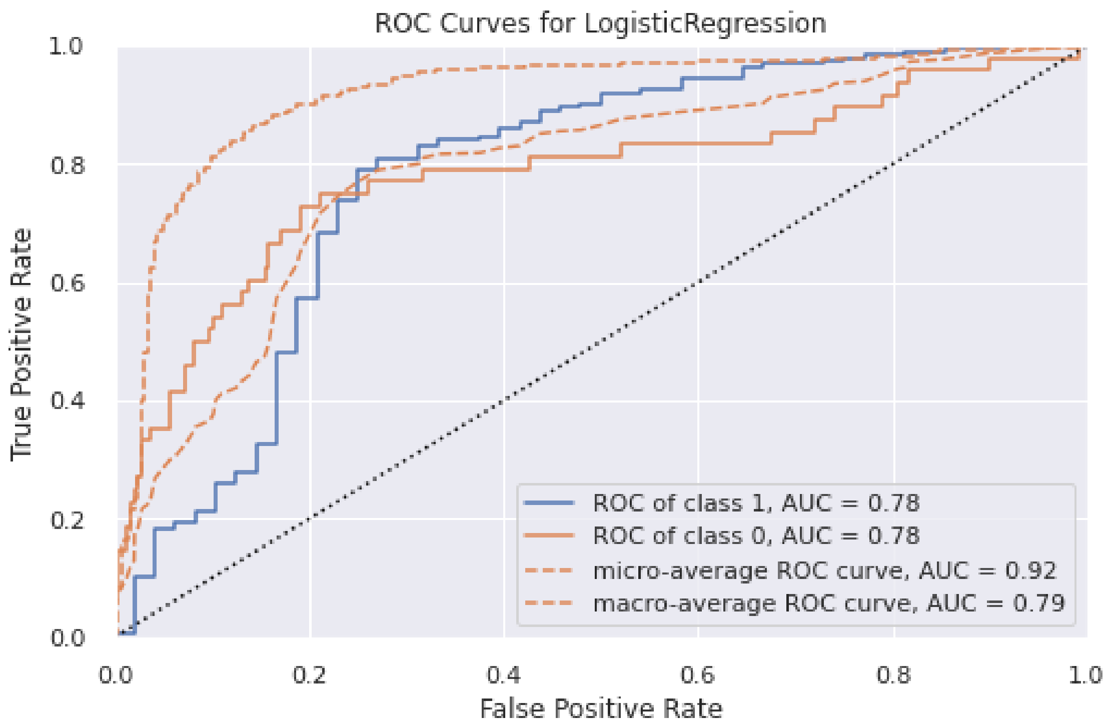
[166]:
```
import numpy as np
from sklearn.metrics import accuracy_score
print("Training accuracy:")
print(np.round(accuracy_score(y_train,clf.predict(x_train)),2))
print("Test accuracy:")
print(np.round(accuracy_score(y_true,y_pred),2))
from sklearn.metrics import confusion_matrix
sn.set(rc={'figure.figsize':(3,3)})
sn.set(font_scale=1)
matrix = confusion_matrix(y_true,y_pred)
sn.heatmap(matrix,annot=True)
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
Training accuracy:
0.86
Test accuracy:
0.86
```

[166]: Text(0.0, 0.5, 'True')



[167]:
```
from yellowbrick.classifier import ROCAUC
visualizer = ROCAUC(clf,classes=[1,0])
sn.set(rc={'figure.figsize':(8,5)})
visualizer.fit(x_train, y_train)
visualizer.score(x_test, y_test)
visualizer.show()
```



[167]: <AxesSubplot:title={'center':'ROC Curves for LogisticRegression'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>

We can see that we're getting a relatively similar result when using all of our original variables. So we were clearly able to reduce dimensionality successfully, reducing from over 30 predictors to 11.

[168]:
```
from sklearn.preprocessing import StandardScaler
y = data_full['Attrition']
x = data_full.drop('Attrition',axis=1)
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

[169]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components = None)
pca.fit(x)
```

[169]: PCA()

[170]:
```
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.api as sm
train = np.random.choice(data.index,200)
train_data = data.loc[pd.Index(train)]
test = np.random.choice(data.index,200)
```

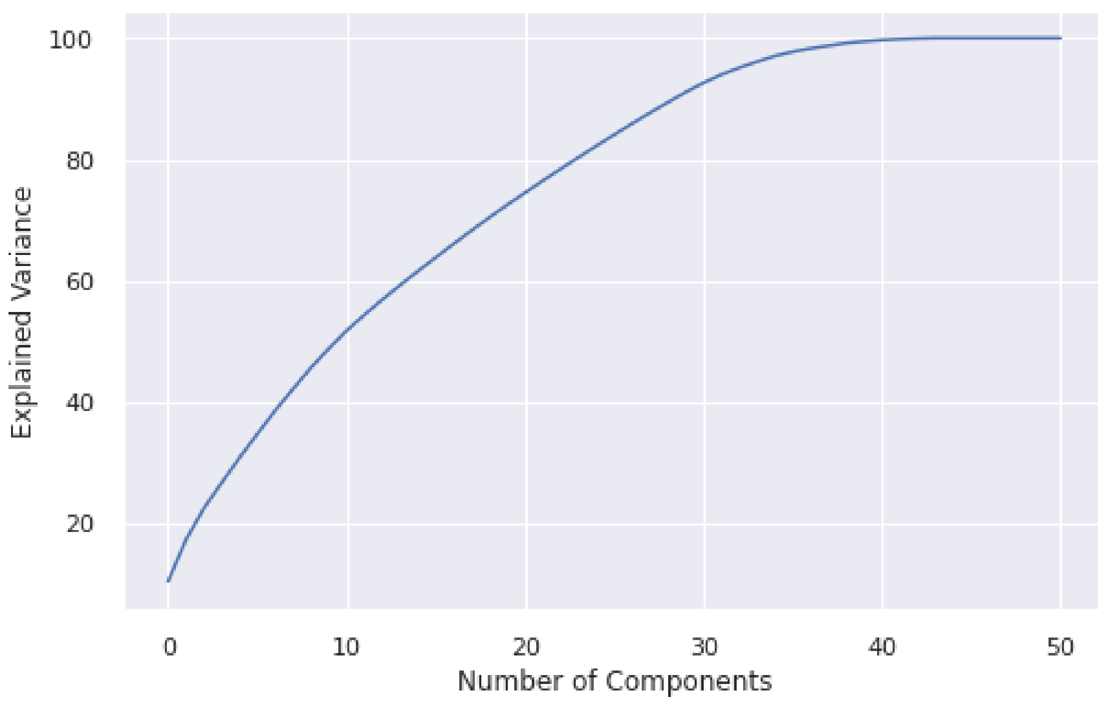
```
test_data = data.loc[pd.Index(train)]
```

[171]:
```
print('Variance Explained - %')
print(pca.explained_variance_ratio_ * 100)
```

```
Variance Explained - %
[1.04948948e+01 6.87358414e+00 5.17043810e+00 4.22547271e+00
 4.12216131e+00 3.92193761e+00 3.83255414e+00 3.54533801e+00
 3.48951017e+00 3.16112386e+00 2.95863529e+00 2.60466287e+00
 2.49698104e+00 2.39279674e+00 2.30537900e+00 2.25217367e+00
 2.24221930e+00 2.19956459e+00 2.14417305e+00 2.07928353e+00
 1.99293944e+00 1.98635845e+00 1.96447115e+00 1.90451280e+00
 1.88158548e+00 1.84621729e+00 1.81914228e+00 1.78006107e+00
 1.71610976e+00 1.68043870e+00 1.54641017e+00 1.34643969e+00
 1.13435417e+00 9.94155254e-01 9.41399841e-01 6.99525497e-01
 5.44454514e-01 4.43135267e-01 4.29131066e-01 2.91281317e-01
 2.17835738e-01 1.58099378e-01 1.01184067e-01 6.78737353e-02
 2.06975304e-30 6.43525977e-31 3.32774468e-31 3.10047628e-31
 2.26411319e-31 1.88038381e-31 6.84064085e-32]
```

[172]:
```
plt.plot(np.cumsum(pca.explained_variance_ratio_*100))
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance')
```

[172]: Text(0, 0.5, 'Explained Variance')



We can see that we're getting relatively similar accuracy, but that the amount of variance we're able to explain tapers off past around 30 variables. So it makes sense why our current predictor set of 11 variables is capturing a good amount of the variance. This is in part because we reduced collinearity manually above, so we would expect to have less variables explaining the overall variance.

We could try to use methods to reduce the dimensionality by combining variables. This will unfortunately make it difficult for us to use the model to infer something about the relationship between our variables and the outcome of attrition. Therefore, we will use another approach to building our model below.

`Recursive Feature Elimination - RFE`

We can use a different method altogether for feature selection - recursive feature elimination. Let's use our original cleaned dataset to see whether this algorithm selects the same features that we did during our EDA.

[173]:
```
from sklearn.feature_selection import RFE
x = data_full.drop(columns=['Attrition'])
y = data_full['Attrition']
train_x, test_x, train_y, test_y = train_test_split(x, y, test_size = 0.2)
model = LogisticRegression()
rfe = RFE(model)
fit = rfe.fit(train_x,train_y)
```

[174]:
```
col = x.columns
RFE_sup = rfe.support_
RFE_rank = rfe.ranking_
dataset = pd.DataFrame({'Columns': col, 'RFE_support': RFE_sup, 'RFE_ranking':␣
 ↪RFE_rank}, columns=['Columns', 'RFE_support', 'RFE_ranking'])
df = dataset[(dataset["RFE_support"] == True) & (dataset["RFE_ranking"] == 1)]
filtered_features = df['Columns']
filtered_features
```

[174]:
```
4               EnvironmentSatisfaction
6                        JobInvolvement
8                       JobSatisfaction
17                TrainingTimesLastYear
18                      WorkLifeBalance
23             BusinessTravel_Non-Travel
24      BusinessTravel_Travel_Frequently
28                     Department_Sales
29         EducationField_Human Resources
30           EducationField_Life Sciences
32                EducationField_Medical
33                  EducationField_Other
34       EducationField_Technical Degree
36                          Gender_Male
```

```
37    JobRole_Healthcare Representative
38              JobRole_Human Resources
39        JobRole_Laboratory Technician
41       JobRole_Manufacturing Director
42            JobRole_Research Director
43           JobRole_Research Scientist
45         JobRole_Sales Representative
46               MaritalStatus_Divorced
48                 MaritalStatus_Single
49                          OverTime_No
50                         OverTime_Yes
Name: Columns, dtype: object
```

[175]:
```python
new_train_x = train_x[filtered_features]
new_test_x = test_x[filtered_features]
```

[176]:
```python
model = sm.Logit(train_y, new_train_x)
model_fit = model.fit()
print(model_fit.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.327630
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                 1176
Model:                          Logit   Df Residuals:                     1151
Method:                           MLE   Df Model:                           24
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.2568
Time:                        21:18:36   Log-Likelihood:                -385.29
converged:                       True   LL-Null:                       -518.44
Covariance Type:            nonrobust   LLR p-value:                 9.499e-43
================================================================================
=====================
                                        coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------------
EnvironmentSatisfaction              -0.4359      0.086     -5.096      0.000
-0.603      -0.268
JobInvolvement                       -0.5570      0.129     -4.305      0.000
-0.811      -0.303
JobSatisfaction                      -0.3936      0.084     -4.675      0.000
-0.559      -0.229
TrainingTimesLastYear                -0.1703      0.079     -2.146      0.032
-0.326      -0.015
WorkLifeBalance                      -0.3382      0.130     -2.602      0.009
```

```
-0.593      -0.083
BusinessTravel_Non-Travel             -0.7742      0.378     -2.047      0.041
-1.515      -0.033
BusinessTravel_Travel_Frequently       0.8275      0.219      3.774      0.000
0.398        1.257
Department_Sales                       0.7175      0.672      1.068      0.286
-0.600       2.035
EducationField_Human Resources         0.6968      0.870      0.801      0.423
-1.009       2.402
EducationField_Life Sciences          -0.2643      0.339     -0.779      0.436
-0.929       0.400
EducationField_Medical                -0.1993      0.360     -0.554      0.580
-0.905       0.506
EducationField_Other                  -0.4178      0.514     -0.813      0.416
-1.425       0.589
EducationField_Technical Degree        0.4553      0.416      1.094      0.274
-0.360       1.271
Gender_Male                            0.3277      0.195      1.677      0.093
-0.055       0.711
JobRole_Healthcare Representative     -0.0786      0.748     -0.105      0.916
-1.544       1.387
JobRole_Human Resources                0.7821      0.807      0.970      0.332
-0.799       2.363
JobRole_Laboratory Technician          1.4457      0.662      2.185      0.029
0.149        2.742
JobRole_Manufacturing Director        -0.2967      0.745     -0.398      0.691
-1.757       1.164
JobRole_Research Director             -1.3119      0.990     -1.326      0.185
-3.252       0.628
JobRole_Research Scientist             0.7273      0.662      1.098      0.272
-0.571       2.025
JobRole_Sales Representative           1.0075      0.366      2.756      0.006
0.291        1.724
MaritalStatus_Divorced                -0.5248      0.286     -1.836      0.066
-1.085       0.035
MaritalStatus_Single                   0.9300      0.203      4.578      0.000
0.532        1.328
OverTime_No                            1.4869      0.991      1.501      0.133
-0.454       3.428
OverTime_Yes                           3.2250      1.007      3.203      0.001
1.251        5.199
================================================================================
=====================
```

[179]:

```
new_train_x = new_train_x.
    ↪drop(columns=['Department_Sales','EducationField_Human␣
    ↪Resources','EducationField_Life␣
    ↪Sciences','EducationField_Medical','EducationField_Other','EducationField_Technical␣
    ↪Degree','Gender_Male','JobRole_Healthcare Representative','JobRole_Human␣
    ↪Resources','JobRole_Manufacturing Director','JobRole_Research␣
    ↪Director','JobRole_Research␣
    ↪Scientist','MaritalStatus_Divorced','OverTime_No'])
new_test_x = new_test_x.drop(columns=['Department_Sales','EducationField_Human␣
    ↪Resources','EducationField_Life␣
    ↪Sciences','EducationField_Medical','EducationField_Other','EducationField_Technical␣
    ↪Degree','Gender_Male','JobRole_Healthcare Representative','JobRole_Human␣
    ↪Resources','JobRole_Manufacturing Director','JobRole_Research␣
    ↪Director','JobRole_Research␣
    ↪Scientist','MaritalStatus_Divorced','OverTime_No'])
```

[180]:
```
model = sm.Logit(train_y, new_train_x)
model_fit = model.fit()
print(model_fit.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.347374
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                 1176
Model:                          Logit   Df Residuals:                     1165
Method:                           MLE   Df Model:                           10
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.2120
Time:                        21:23:47   Log-Likelihood:                -408.51
converged:                       True   LL-Null:                       -518.44
Covariance Type:            nonrobust   LLR p-value:                 1.143e-41
================================================================================
====================
                                       coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
--------------------
EnvironmentSatisfaction             -0.3453      0.078     -4.449      0.000
-0.497      -0.193
JobInvolvement                      -0.3404      0.104     -3.268      0.001
-0.545      -0.136
JobSatisfaction                     -0.2860      0.075     -3.823      0.000
-0.433      -0.139
TrainingTimesLastYear               -0.1005      0.071     -1.421      0.155
-0.239       0.038
WorkLifeBalance                     -0.1256      0.106     -1.190      0.234
```

```
-0.332      0.081
BusinessTravel_Non-Travel          -0.6962      0.369     -1.887      0.059
-1.419      0.027
BusinessTravel_Travel_Frequently    0.7041      0.209      3.366      0.001
0.294      1.114
JobRole_Laboratory Technician       1.0009      0.217      4.620      0.000
0.576      1.426
JobRole_Sales Representative        1.3066      0.325      4.021      0.000
0.670      1.943
MaritalStatus_Single                1.1158      0.182      6.137      0.000
0.759      1.472
OverTime_Yes                        1.6679      0.187      8.926      0.000
1.302      2.034
================================================================================
====================
```

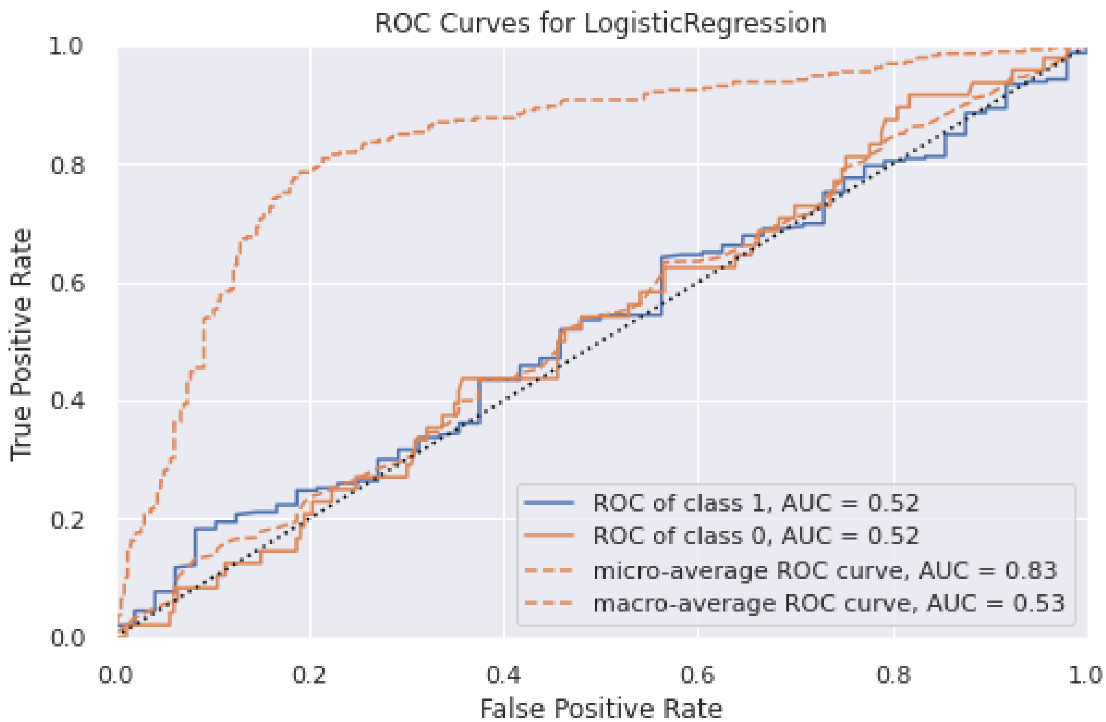
[181]:

```
clf = LogisticRegression(max_iter = 2500)
clf.fit(new_train_x,train_y)
print(clf.coef_,clf.intercept_)
y_pred = clf.predict(new_test_x)
y_true = y_test
```

```
[[-0.41243144 -0.53619907 -0.37154746 -0.17002249 -0.3276776  -0.70095568
   0.66775292  0.89932744  1.18899747  1.04214345  1.5653859 ]] [1.79954153]
```

[182]:

```
from yellowbrick.classifier import ROCAUC
visualizer = ROCAUC(clf,classes=[1,0])
sn.set(rc={'figure.figsize':(8,5)})
visualizer.fit(new_train_x, train_y)
visualizer.score(new_test_x, y_test)
visualizer.show()
```



[182]: <AxesSubplot:title={'center':'ROC Curves for LogisticRegression'}, xlabel='False Positive Rate', ylabel='True Positive Rate'>

Oversampling: SMOTE

[183]:
```
from imblearn.over_sampling import SMOTE
columns_x =columns_x = new_train_x.columns
sm = SMOTE(random_state=0)
trainX_sm ,trainY_sm = sm.fit_resample(new_train_x, train_y)

train_x_smote = pd.DataFrame(data=trainX_sm,columns=columns_x)
train_y_smote = pd.DataFrame(data=trainY_sm,columns=['Attrition'])
```

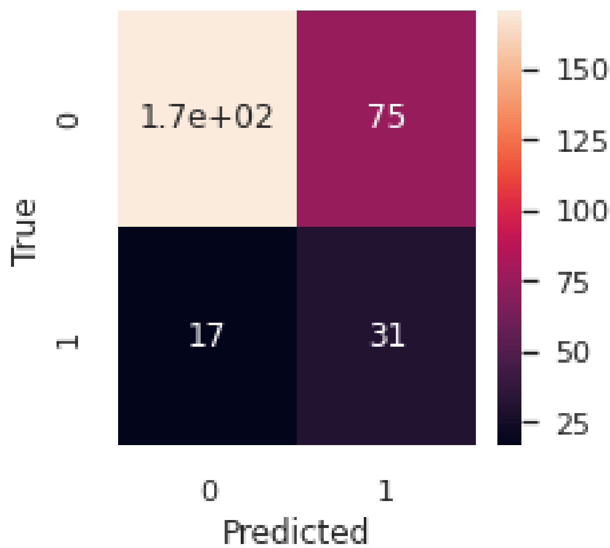
[184]:
```
trainX = train_x_smote
testX = new_test_x
trainY = train_y_smote
testY = test_y

logreg = LogisticRegression()
logreg.fit(trainX, trainY)

y_pr = logreg.predict(testX)
```

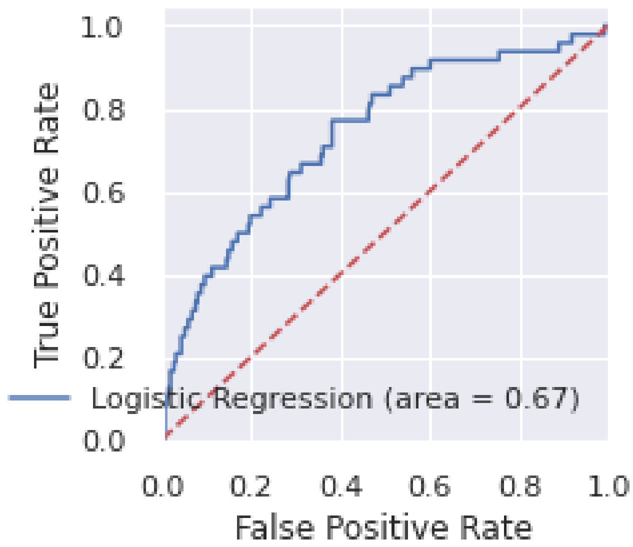
[185]:
```
sn.set(rc={'figure.figsize':(3,3)})
sn.set(font_scale=1)
matrix = confusion_matrix(testY,y_pr)
sn.heatmap(matrix,annot=True)
plt.xlabel('Predicted')
plt.ylabel('True')
print("Training accuracy:")
print(np.round(accuracy_score(trainY,logreg.predict(trainX)),2))
print("Test accuracy:")
print(np.round(accuracy_score(testY,y_pr),2))
```

```
Training accuracy:
0.72
Test accuracy:
0.69
```



[189]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(testY, logreg.predict(testX))
fpr, tpr, thresholds = roc_curve(testY, logreg.predict_proba(testX)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
```

```
plt.show()
```



Although we used a redundant factor elimination method, we are seeing worse performance. This is likely because, as we saw in our principle component analysis, we have many variables that contribute to the variance.

[190]:
```
y = data2['Attrition']
x = data2.drop('Attrition',axis=1)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.20,␣
↪shuffle=True, random_state=2)
```

[191]:
```
from imblearn.over_sampling import SMOTE
columns_x = x_train.columns
sm = SMOTE(random_state=0)
trainX_smote ,trainY_smote = sm.fit_resample(x_train, y_train)

train_x_smote = pd.DataFrame(data=trainX_smote,columns=columns_x)
train_y_smote = pd.DataFrame(data=trainY_smote,columns=['Attrition'])
```

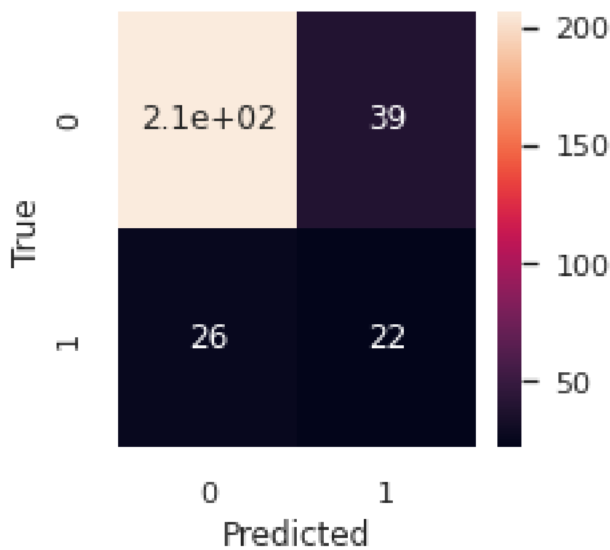
[192]:
```
trainX= train_x_smote
testX = x_test
trainY = train_y_smote
testY = y_test

logreg = LogisticRegression()
logreg.fit(trainX, trainY)
```

```
y_pred = logreg.predict(testX)
```

[193]:
```
sn.set(rc={'figure.figsize':(3,3)})
sn.set(font_scale=1)
matrix = confusion_matrix(testY,y_pred)
sn.heatmap(matrix,annot=True)
plt.xlabel('Predicted')
plt.ylabel('True')
print("Training accuracy:")
print(np.round(accuracy_score(trainY,logreg.predict(trainX)),2))
print("Test accuracy:")
print(np.round(accuracy_score(testY,y_pred),2))
```

```
Training accuracy:
0.8
Test accuracy:
0.78
```

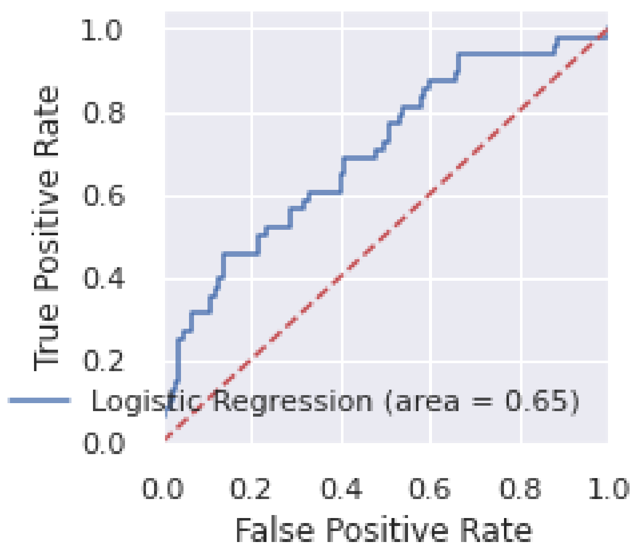


We can see that we have lower prediction accuracy, and

[195]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(testY, logreg.predict(testX))
fpr, tpr, thresholds = roc_curve(testY, logreg.predict_proba(testX)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
```

```
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")

plt.show()
```



We can see that performance for this model is even worse - our area under the curve is only 0.65. We can interpret this to mean relatively poor performance of our model, using the oversampling technique. This could mean that we were overfitting the data with our first model, or that there are other models that would be a better fit.

`Forward Stepwise Refinement`

We've seen with our principal component analysis that we're able to, with relative accuracy, use a logistic regression model with our current predictor set to predict attrition.

We could explore more flexible models, but because the intent is for business leaders to understand attrition, and make decisions based on the outcomes, we will stick to simpler approaches, so that the models are easier to interpret and make sense of.

Unfortunately, it appears that we need all of the variables in our current predictor set to capture enough of the outcome variance. If we're unable to leverage a few variables to create a logistic regression model, we can explore creating new features, although this will reduce the interpretability of the model.

Because we want to determine the variables that are most strongly predictive of attrition, we will use forward stepwise refinement to identify how many variables we should add to maximize our log-likelihood score.

[196]:
```python
allowed_factors = data.columns.values.tolist()
allowed_factors = allowed_factors[1:]
print("Allowed Factors: ",allowed_factors)
```

```
Allowed Factors:  ['DailyRate', 'DistanceFromHome', 'EnvironmentSatisfaction',
'GenderMale', 'JobInvolvement', 'JobLevel', 'JobSatisfaction',
'NumCompaniesWorked', 'OverTime', 'RelationshipSatisfaction', 'WorkLifeBalance']
```

[197]:
```python
#First step: create training data

import numpy as np
import statsmodels.formula.api as smf
import statsmodels.api as sm

train = np.random.choice(data.index,200)
train_data = data.loc[pd.Index(train)]

test = np.random.choice(data.index,200)
test_data = data.loc[pd.Index(train)]
```

[198]:
```python
#Second step: start creating the model

#p = 1
model_1 = smf.logit(formula='Attrition~DistanceFromHome',data=train_data).
 ↪fit(maxiter=35,disp=0)
max_val = model_1.llf
best_item = 'DistanceFromHome'

for item in allowed_factors:
    string = 'Attrition~'
    string = string + item
    #print(item)
    model_1 = smf.logit(formula=string,data=train_data).fit(maxiter=100,disp=0)
    val = model_1.llf
    #print(val)
    if val > max_val:
        max_val = val
        best_item = item

str_1 = 'Attrition~' + best_item
model_fin = smf.logit(formula=str_1,data=train_data).fit(maxiter=35,disp=0)
val1 = model_fin.llf
print(best_item, ': ', val1)
```

```
OverTime :  -81.69714624342696
```

```
[199]: model_fin.summary()
```

```
[199]: <class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      198
Method:                           MLE   Df Model:                            1
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.07093
Time:                        21:26:22   Log-Likelihood:                -81.697
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                 0.0004127
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -2.1748      0.272     -7.981      0.000      -2.709      -1.641
OverTime       1.4244      0.401      3.552      0.000       0.638       2.210
==============================================================================
"""
```

We see that Environment Satisfaction was chosen as the best model with one predictor. Let's try with two variables and see whether we can improve our model.

```
[200]: #p = 2
allowed_factors.remove(best_item)
str_add = 'Attrition~' + best_item + '+' + allowed_factors[0]
model_2 = smf.logit(formula=str_add,data=train_data).fit(maxiter=35,disp=0)
max_val2 = model_2.llf
best_item_2 = allowed_factors[0]
for item in allowed_factors:
    string = 'Attrition~' + best_item
    string = string + '+' + item
    #print(item)
    model_2 = smf.logit(formula=string,data=train_data).fit(maxiter=100,disp=0)
    val = model_2.llf
    #print(val)
    if val > max_val2:
        max_val2 = val
        best_item_2 = item
str_2 = str_1 + '+' + best_item_2
model_fin_2 = smf.logit(formula=str_2,data=train_data).fit(maxiter=35,disp=0)
val2 = model_fin_2.llf
print(best_item_2, ': ', val2)
```

```
JobLevel :  -76.77482772703748
```

```
[201]: model_fin_2.summary()
```

[201]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      197
Method:                           MLE   Df Model:                            2
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.1269
Time:                        21:26:23   Log-Likelihood:                -76.775
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                 1.424e-05
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.7862      0.525     -1.497      0.134      -1.815       0.243
OverTime       1.4684      0.415      3.537      0.000       0.655       2.282
JobLevel      -0.7501      0.273     -2.744      0.006      -1.286      -0.214
==============================================================================
"""
```

[202]:
```python
#p = 3
allowed_factors.remove(best_item_2)
str_add = 'Attrition~' + best_item + '+' + best_item_2 + '+' +␣
 ↪allowed_factors[0]
model_3 = smf.logit(formula=str_add,data=train_data).fit(maxiter=35,disp=0)
max_val3 = model_3.llf
best_item_3 = allowed_factors[0]
for item in allowed_factors:
    string = 'Attrition~' + best_item + '+' + best_item_2 + '+'+ item
    #print(item)
    model_3 = smf.logit(formula=string,data=train_data).fit(maxiter=100,disp=0)
    val = model_3.llf
    #print(val)
    if val > max_val3:
        max_val3 = val
        best_item_3 = item
str_3 = str_2 + '+' + best_item_3
model_fin_3 = smf.logit(formula=str_3,data=train_data).fit(maxiter=35,disp=0)
val3 = model_fin_3.llf
print(best_item_3, ': ', val3)
```

```
NumCompaniesWorked :  -72.34742745970087
```

[203]:
```python
model_fin_3.summary()
```

[203]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      196
Method:                           MLE   Df Model:                            3
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.1773
Time:                        21:26:24   Log-Likelihood:                -72.347
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                 7.816e-07
================================================================================
======
                         coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept             -1.4147      0.602     -2.350      0.019      -2.595
-0.235
OverTime               1.6535      0.441      3.750      0.000       0.789
2.518
JobLevel              -0.8182      0.287     -2.854      0.004      -1.380
-0.256
NumCompaniesWorked     0.2293      0.076      3.007      0.003       0.080
0.379
================================================================================
======
"""
```

[204]:

```
#p = 4
allowed_factors.remove(best_item_3)
str_add = 'Attrition~' + best_item + '+' + best_item_2 + '+' + best_item_3 +
 ↪'+' + allowed_factors[0]
model_4 = smf.logit(formula=str_add,data=train_data).fit(maxiter=35,disp=0)
max_val4 = model_4.llf
best_item_4 = allowed_factors[0]
for item in allowed_factors:
    string = 'Attrition~' + best_item + '+' + best_item_2 + '+' + best_item_3 +
 ↪'+' + item
    #print(item)
    model_4 = smf.logit(formula=string,data=train_data).fit(maxiter=100,disp=0)
    val = model_4.llf
    #print(val)
    if val > max_val4:
        max_val4 = val
        best_item_4 = item
str_4 = str_3 + '+' + best_item_4
model_fin_4 = smf.logit(formula=str_4,data=train_data).fit(maxiter=35,disp=0)
val4 = model_fin_4.llf
```

```python
print(best_item_4, ': ', val4)
```

```
JobInvolvement :  -68.75048860547048
```

[205]:
```python
model_fin_4.summary()
```

[205]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      195
Method:                           MLE   Df Model:                            4
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.2182
Time:                        21:26:26   Log-Likelihood:                -68.750
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                 9.413e-08
================================================================================
======
                         coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept             -0.6099      0.691     -0.882      0.378      -1.965
0.745
OverTime               1.6434      0.455      3.609      0.000       0.751
2.536
JobLevel              -0.8844      0.306     -2.889      0.004      -1.484
-0.284
NumCompaniesWorked     0.2485      0.079      3.143      0.002       0.094
0.403
JobInvolvement        -1.1944      0.448     -2.667      0.008      -2.072
-0.317
================================================================================
======
"""
```

Based on the p-values of the intercepts, we can see that our model improves as we add more features. However, with a very low r-squared value, we are still not creating a model that is a good fit to our data.

Let's see if we add all of our variables that we found explained part of the variance in the PCA step of our analysis if we get better accuracy

[206]:
```python
factors = data.columns.values.tolist()
factors = factors[1:]

formula = ''
```

```
for item in range(len(factors)-1):
    formula = formula + factors[item] + '+'
formula = formula + factors[-1]
final_formula = 'Attrition~' + formula
print(formula)
```

```
DailyRate+DistanceFromHome+EnvironmentSatisfaction+GenderMale+JobInvolvement+Job
Level+JobSatisfaction+NumCompaniesWorked+OverTime+RelationshipSatisfaction+WorkL
ifeBalance
```

[207]:
```
model_1 = smf.logit(formula=final_formula,data=train_data).
 ↪fit(maxiter=35,disp=0)
model_1.summary()
```

[207]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      188
Method:                           MLE   Df Model:                           11
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.2786
Time:                        21:26:27   Log-Likelihood:                -63.434
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                 9.462e-07
================================================================================
============
                              coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
------------
Intercept                  -0.1068      1.129     -0.095      0.925      -2.320
2.107
DailyRate                  -0.0004      0.001     -0.664      0.507      -0.002
0.001
DistanceFromHome            0.0540      0.030      1.777      0.076      -0.006
0.114
EnvironmentSatisfaction    -0.5697      0.477     -1.194      0.232      -1.505
0.365
GenderMale                  0.3881      0.517      0.751      0.453      -0.625
1.401
JobInvolvement             -1.0683      0.484     -2.209      0.027      -2.016
-0.121
JobLevel                   -0.9397      0.320     -2.936      0.003      -1.567
-0.312
JobSatisfaction            -0.7779      0.470     -1.656      0.098      -1.699
```

```
0.143
NumCompaniesWorked           0.2805      0.087      3.221      0.001      0.110
0.451
OverTime                     1.6687      0.491      3.397      0.001      0.706
2.632
RelationshipSatisfaction    -0.5748      0.486     -1.184      0.237     -1.527
0.377
WorkLifeBalance             -0.0710      0.497     -0.143      0.886     -1.045
0.903
================================================================================
============
"""
```

[208]:

```
model_1 = smf.logit(formula='Attrition~OverTime*JobLevel + JobLevel +␣
 ↪OverTime',data=train_data).fit(maxiter=35,disp=0)
model_1.summary()
```

[208]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      196
Method:                           MLE   Df Model:                            3
Date:                Mon, 02 May 2022   Pseudo R-squ.:                  0.1348
Time:                        21:26:27   Log-Likelihood:                -76.081
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                 2.876e-05
================================================================================
=====
                       coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept           -1.2268      0.641     -1.914      0.056      -2.483
0.030
OverTime             2.6187      1.087      2.410      0.016       0.489
4.749
JobLevel            -0.4902      0.328     -1.494      0.135      -1.133
0.153
OverTime:JobLevel   -0.6707      0.586     -1.145      0.252      -1.818
0.477
================================================================================
=====
"""
```

```
[209]: model_1 = smf.logit(formula='Attrition~OverTime*EnvironmentSatisfaction +␣
        ↪EnvironmentSatisfaction + OverTime',data=train_data).fit(maxiter=35,disp=0)
       model_1.summary()
```

```
[209]: <class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      196
Method:                           MLE   Df Model:                            3
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.08963
Time:                        21:26:28   Log-Likelihood:                -80.053
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                  0.001268
================================================================================
====================
                                       coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
--------------------
Intercept                           -1.6529      0.364     -4.543      0.000
-2.366      -0.940
OverTime                             0.8797      0.613      1.435      0.151
-0.322       2.082
EnvironmentSatisfaction             -0.9980      0.558     -1.790      0.073
-2.091       0.095
OverTime:EnvironmentSatisfaction     1.0336      0.830      1.245      0.213
-0.593       2.660
================================================================================
====================
"""
```

Doing this manually is unlikely to be helpful. Let's see if we can investigate how to create all combimations of our predictors, to see which are most influential.

We can try to add quadratic terms, to see if we can improve our model. However, it is clear that we are not able to improve fit by adding interaction terms.

Despite investigating interaction effects, we are still not generating a very good fit to our data.

However, we do gain valuable information from the coefficients that can help us infer some information about attrition and our data.

If we are looking to perform better on predicting outcomes, we can explore non-parametric methods, like k-nearest neighbors.

We can also leverage a different approach to variable selection, to see if we can create a better model.

```
Domain Knowledge & Business Questions
```

We have not been successful in modeling attrition using variable selection methods like forwards stepwise selection. But because our goal is to provide inference about what variables are associated with attrition, we can also use our domain knowledge and try modeling with a few features of interest to our leaders.

> A common trend is that individuals with less tenure or less experience tend to leave more often than those that have more tenure / more experience.

Let's try modeling attrition with variables that capture tenure, experience and age, to see whether these predictors are strongly associated with attrition. This is an effort to optimize for inference, rather than prediction, so we accept the bias and variance that might result from selecting variables based on domain knowledge and hypotheses, rather than selecting the features that are most strongly associated with a higher prediction accuracy.

[210]:
```
train_age = np.random.choice(data2.index,200)
train_data_age = data2.loc[pd.Index(train)]

test_age = np.random.choice(data2.index,200)
test_data_age = data2.loc[pd.Index(train)]
```

[211]:
```
model_1 = smf.logit(formula='Attrition~Age',data=train_data_age).fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.432163
         Iterations 6
```

[211]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      198
Method:                           MLE   Df Model:                            1
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.01707
Time:                        21:26:31   Log-Likelihood:                -86.433
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                   0.08312
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.2522      0.839     -0.301      0.764      -1.897       1.392
Age           -0.0397      0.024     -1.675      0.094      -0.086       0.007
==============================================================================
"""
```

[212]:
```python
model_1 = smf.logit(formula='Attrition~JobLevel', data = train_data_age).
  ↪fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.415163
         Iterations 7
```

[212]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      198
Method:                           MLE   Df Model:                            1
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.05574
Time:                        21:26:31   Log-Likelihood:                -83.033
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                  0.001743
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.3146      0.481     -0.654      0.513      -1.257       0.628
JobLevel      -0.7160      0.260     -2.752      0.006      -1.226      -0.206
==============================================================================
"""
```

[213]:
```python
model_1 = smf.logit(formula='Attrition~JobLevel + Age', data = train_data_age).
  ↪fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.414917
         Iterations 7
```

[213]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      197
Method:                           MLE   Df Model:                            2
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.05630
Time:                        21:26:32   Log-Likelihood:                -82.983
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                  0.007080
==============================================================================
```

```
                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0904      0.864     -0.105      0.917      -1.783       1.602
JobLevel      -0.6830      0.281     -2.431      0.015      -1.234      -0.132
Age           -0.0081      0.026     -0.312      0.755      -0.059       0.043
==============================================================================
"""
```

[214]:
```
model_1 = smf.logit(formula='Attrition~JobLevel * Age + JobLevel+Age', data =
    train_data_age).fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.409435
         Iterations 8
```

[214]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      196
Method:                           MLE   Df Model:                            3
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.06877
Time:                        21:26:32   Log-Likelihood:                -81.887
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                  0.007068
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       -3.2005      2.405     -1.331      0.183      -7.914       1.513
JobLevel         1.2205      1.402      0.870      0.384      -1.528       3.969
Age              0.0786      0.068      1.148      0.251      -0.056       0.213
JobLevel:Age    -0.0517      0.039     -1.337      0.181      -0.128       0.024
================================================================================
"""
```

We can see that investigating age and job level predictors does not yield a good classifier - therefore, we would likely share with leaders that there are potentially other more significant factors we could consider, and that the relationship is not as simple as Attrition~Age

Let's investigate another question that is commonly asked, to see if we find any helpful information about features of interest.

> It is a common result that employees who are less engaged, and less happy are more likely to leave.

Let's see if it possible to create a simple classifier just with this feature.

[215]:
```
train_sat = np.random.choice(data2.index,200)
train_data_sat = data2.loc[pd.Index(train)]

test_sat = np.random.choice(data2.index,200)
test_data_sat = data2.loc[pd.Index(train)]
```

[216]:
```
model_1 = smf.logit(formula='Attrition~RelationshipSatisfaction', data =␣
↪train_data_age).fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.433482
         Iterations 6
```

[216]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      198
Method:                           MLE   Df Model:                            1
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.01407
Time:                        21:26:34   Log-Likelihood:                -86.696
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                    0.1157
================================================================================
============
                               coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
------------
Intercept                   -1.3049      0.282     -4.630      0.000      -1.857
-0.752
RelationshipSatisfaction    -0.6138      0.389     -1.579      0.114      -1.376
0.148
================================================================================
============
"""
```

[217]:
```
model_1 = smf.logit(formula='Attrition~JobInvolvement', data = train_data_age).
↪fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.423669
         Iterations 6
```

[217]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      198
Method:                           MLE   Df Model:                            1
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.03639
Time:                        21:26:34   Log-Likelihood:                -84.734
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                   0.01141
================================================================================
==
                    coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
--
Intercept        -1.0341      0.291     -3.553      0.000      -1.605
-0.464
JobInvolvement   -1.0055      0.394     -2.551      0.011      -1.778
-0.233
================================================================================
==
"""
```

[218]:
```
model_1 = smf.logit(formula='Attrition~EnvironmentSatisfaction', data =␣
    ↪train_data_age).fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.436173
         Iterations 6
```

[218]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      198
Method:                           MLE   Df Model:                            1
Date:                Mon, 02 May 2022   Pseudo R-squ.:                0.007953
Time:                        21:26:35   Log-Likelihood:                -87.235
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                    0.2369
================================================================================
===========
                             coef    std err          z      P>|z|      [0.025
```

```
0.975]
------------------------------------------------------------------------------------
-----------
Intercept                   -1.3863      0.289     -4.802      0.000      -1.952
-0.821
EnvironmentSatisfaction     -0.4626      0.389     -1.189      0.234      -1.225
0.300
====================================================================================
===========
"""
```

[219]:
```
model_1 = smf.logit(formula='Attrition~WorkLifeBalance', data = train_data_age).
 ↪fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.439602
         Iterations 6
```

[219]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      198
Method:                           MLE   Df Model:                            1
Date:                Mon, 02 May 2022   Pseudo R-squ.:               0.0001551
Time:                        21:26:35   Log-Likelihood:                -87.920
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                    0.8688
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept          -1.7047      0.344     -4.959      0.000      -2.379
-1.031
WorkLifeBalance     0.0684      0.415      0.165      0.869      -0.746
0.882
================================================================================
===
"""
```

It appears that any of these variables on their own is not a strong predictor of attrition. Let's investigate interaction effects, to see if we can create a stronger model.

[220]:
```
model_1 = smf.
 ↪logit(formula='Attrition~WorkLifeBalance*EnvironmentSatisfaction+WorkLifeBalance+Environmen
 ↪data = train_data_age).fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.429205
         Iterations 6
```

[220]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      196
Method:                           MLE   Df Model:                            3
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.02380
Time:                        21:26:36   Log-Likelihood:                -85.841
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                    0.2421
================================================================================
===========================
                                              coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
---------------------------
Intercept                                  -2.0369      0.614     -3.318
0.001      -3.240      -0.834
WorkLifeBalance                             0.9109      0.698      1.305
0.192      -0.457       2.279
EnvironmentSatisfaction                     0.5171      0.742      0.697
0.486      -0.938       1.972
WorkLifeBalance:EnvironmentSatisfaction    -1.4192      0.880     -1.613
0.107      -3.144       0.306
================================================================================
===========================
"""
```

[221]:
```
model_1 = smf.
 ↪logit(formula='Attrition~WorkLifeBalance*JobInvolvement+JobInvolvement+WorkLifeBalance',␣
 ↪data = train_data_age).fit(maxiter=35)
model_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.423585
         Iterations 6
```

[221]: <class 'statsmodels.iolib.summary.Summary'>

```
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Attrition   No. Observations:                  200
Model:                          Logit   Df Residuals:                      196
Method:                           MLE   Df Model:                            3
Date:                Mon, 02 May 2022   Pseudo R-squ.:                 0.03658
Time:                        21:26:37   Log-Likelihood:                -84.717
converged:                       True   LL-Null:                       -87.934
Covariance Type:            nonrobust   LLR p-value:                   0.09230
================================================================================
==================
                                   coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
Intercept                       -1.0986      0.516     -2.127      0.033
-2.111      -0.086
WorkLifeBalance                  0.0953      0.625      0.152      0.879
-1.130       1.321
JobInvolvement                  -0.9808      0.701     -1.399      0.162
-2.355       0.393
WorkLifeBalance:JobInvolvement  -0.0368      0.848     -0.043      0.965
-1.699       1.625
================================================================================
==================
"""
```

Discussion & Results

We are clearly unable to create a good fit to our data using Forward Stepwise Selection and domain knowledge alone, which suggests one of three things: 1. These variables are not good predictors of attrition for our dataset 2. The decision boundary for our classifier is non-linear, and trying to use a simple model like logistic regression analysis will not yield good results. 3. There is a complicated relationship between the predictors, including interaction effects and quadractic terms, which we can't iterate through manually, given how many predictors we suspect might be needed to create the model.

Our results also tell us something that businesses already know - attrition is usually not as simple as paying someone more, or keeping them engaged - it is a highly personal, flexible combination of factors that are associated with attrition, and a simple logistic regression model is unlikely to capture the complexity of this problem completely.

We can use logistic regression to create a relatively accurate model, but if we want to infer meaning from coefficients, we would need to accept that the model will be relatively complex.

Our best models had high dimensionality, and we did not always select the same variables with different approaches. This suggests that it is likely we might be overfitting the data, or that there is not a strong set of predictors available in this dataset. Future investigations should explore the impact of overfitting, additional features, and other less parametric approaches, to see if it is possible to create a more robust, better performing model.

```
[ ]:
```