Title: "Factors That May Lead To Job Attrition" -Data Analysis and Random Forest Classifications

---

The Structure of The Report:

The Table of Contents listed in the following section provides the general structure of this report. Motivation, Exploratory Questions, and Data Source are fully addressed before the report shows any code and/or visualizations. For Methodologies, Analysis, and Results are found in detail with the main content of this report together with accompanying code and visualizations. The section of conclusion addresses the summary conclusions before the main content, where coding/visualization begins.

Table of Contents:

(The Coding/visualization contents starts here)

---

[1]
```
import pandas as pd # linear algebra
import numpy as np  # data processing, CSV file I/O (e.g. pd.read_csv)
import scipy as sp
import sklearn as sk
import seaborn as sns # visualization
from matplotlib.pyplot import suptitle
from seaborn import set
import matplotlib.pyplot as plt # for plotting
from statsmodels.graphics.mosaicplot import mosaic

from matplotlib import colors as mcolors
from pandas import DataFrame, read_html
%matplotlib inline
from scipy import stats # statistical analysis
import scipy.stats as ss # statistical analysis
```

```
from scipy.stats import chisquare

sns.set()

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClas
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (accuracy_score, log_loss, classification_rep
#from imblearn.over_sampling import SMOTE
#import xgboost

# Import statements required for Plotly
#import plotly.offline as py
#py.init_notebook_mode(connected=True)
#import plotly.graph_objs as go
#import plotly.tools as tls

# Import and suppress warnings
import warnings
warnings.filterwarnings('ignore')


# For clustering:
from scipy.cluster.hierarchy import dendrogram
import scipy.cluster.hierarchy as shc
from scipy.spatial.distance import cdist

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering


# For classifications:
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
import sklearn.ensemble as skens
import sklearn.metrics as skmetric
import sklearn.naive_bayes as sknb
import sklearn.tree as sktree
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(style='white', color_codes=True, font_scale=1.3)
import sklearn.externals.six as sksix
import IPython.display as ipd
from sklearn.model_selection import cross_val_score
from sklearn import metrics
import os
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import GaussianNB

#Do I need these (for decision trees):
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
#import pydotplus

# K-Means
from sklearn.cluster import KMeans
import sklearn as sk
from sklearn import metrics
from scipy.spatial.distance import cdist
import numpy as np
import matplotlib.pyplot as plt


# Dendrogram
from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt


from sklearn.cluster import AgglomerativeClustering

from sklearn.model_selection import train_test_split

import sklearn.decomposition as skd
import sklearn.preprocessing as skp
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d import proj3d
```

[2]
```
attrition_data = pd.read_csv('/Users/yukolopez/Practice/SI618Works/FinalP
```

[3]
```
# Convert EmployeeNumber colum to DataFrame index:
attrition_data.set_index('EmployeeNumber', inplace=True)
```

[4]
```
attrition_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1470 entries, 1 to 2068
Data columns (total 34 columns):
Age                        1470 non-null int64
Attrition                  1470 non-null object
BusinessTravel             1470 non-null object
DailyRate                  1470 non-null int64
Department                 1470 non-null object
DistanceFromHome           1470 non-null int64
Education                  1470 non-null int64
EducationField             1470 non-null object
```

```
EmployeeCount                1470 non-null int64
EnvironmentSatisfaction      1470 non-null int64
Gender                       1470 non-null object
HourlyRate                   1470 non-null int64
JobInvolvement               1470 non-null int64
JobLevel                     1470 non-null int64
JobRole                      1470 non-null object
JobSatisfaction              1470 non-null int64
MaritalStatus                1470 non-null object
MonthlyIncome                1470 non-null int64
MonthlyRate                  1470 non-null int64
NumCompaniesWorked           1470 non-null int64
Over18                       1470 non-null object
OverTime                     1470 non-null object
PercentSalaryHike            1470 non-null int64
PerformanceRating            1470 non-null int64
RelationshipSatisfaction     1470 non-null int64
StandardHours                1470 non-null int64
StockOptionLevel             1470 non-null int64
TotalWorkingYears            1470 non-null int64
TrainingTimesLastYear        1470 non-null int64
```

[5]
```
attrition_data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Departr
EmployeeNumber					
1	41	Yes	Travel_Rarely	1102	Sales
2	49	No	Travel_Frequently	279	Researcl Develop
4	37	Yes	Travel_Rarely	1373	Researcl Develop
5	33	No	Travel_Frequently	1392	Researcl Develop
7	27	No	Travel_Rarely	591	Researcl Develop

5 rows × 34 columns

[6]
```
# making sure no spaces before/after each column names.
attrition_data.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
       'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'Over18',
       'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
```

```
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

[7]
```
# checking to see if missing values exist:
attrition_data.isnull().sum(axis = 0)
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               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
```

[8]
```
attrition_data.describe()
```

	Age	DailyRate	DistanceFromHome	Education	Emp
count	1470.000000	1470.000000	1470.000000	1470.000000	147
mean	36.923810	802.485714	9.192517	2.912925	1.0
std	9.135373	403.509100	8.106864	1.024165	0.0
min	18.000000	102.000000	1.000000	1.000000	1.0
25%	30.000000	465.000000	2.000000	2.000000	1.0
50%	36.000000	802.000000	7.000000	3.000000	1.0

	Age	DailyRate	DistanceFromHome	Education	Emp
75%	43.000000	1157.000000	14.000000	4.000000	1.0
max	60.000000	1499.000000	29.000000	5.000000	1.0

8 rows × 25 columns

.unique() method is applied for the columns that appear to be categorical rather than numeric.

[9]
```
attrition_data['BusinessTravel'].unique()
```

```
array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)
```

[10]
```
attrition_data['Department'].unique()
```

```
array(['Sales', 'Research & Development', 'Human Resources'], dtype=object)
```

[11]
```
attrition_data['EducationField'].unique()
```

```
array(['Life Sciences', 'Other', 'Medical', 'Marketing',
       'Technical Degree', 'Human Resources'], dtype=object)
```

[12]
```
attrition_data['EducationField'].unique()
```

```
array(['Life Sciences', 'Other', 'Medical', 'Marketing',
       'Technical Degree', 'Human Resources'], dtype=object)
```

[13]
```
attrition_data['JobLevel'].unique()
```

```
array([2, 1, 3, 4, 5])
```

[14]
```
attrition_data['JobRole'].unique()
```

```
array(['Sales Executive', 'Research Scientist', 'Laboratory Technician',
       'Manufacturing Director', 'Healthcare Representative', 'Manager',
       'Sales Representative', 'Research Director', 'Human Resources'],
      dtype=object)
```

[15]
```
attrition_data['MaritalStatus'].unique()
```

```
array(['Single', 'Married', 'Divorced'], dtype=object)
```

[16]
```
attrition_data['Over18'].unique()
```

```
array(['Y'], dtype=object)
```

[17]
```
attrition_data['OverTime'].unique()
```

```
array(['Yes', 'No'], dtype=object)
```

[18]
```
attrition_data['StandardHours'].unique()
```

```
array([80])
```

[19]
```
attrition_data['StockOptionLevel'].unique()
```

```
array([0, 1, 3, 2])
```

[20]
```
attrition_data['TotalWorkingYears'].unique().min()
```

```
0
```

[21]
```
attrition_data['TotalWorkingYears'].unique().max()
```

```
40
```

[22]
```
attrition_data['TrainingTimesLastYear'].unique()
```

```
array([0, 3, 2, 5, 1, 4, 6])
```

[23]
```
attrition_data['YearsAtCompany'].unique().min()
```

```
0
```

[24]
```
attrition_data['YearsAtCompany'].unique().max()
```

```
40
```

```
[25] attrition_data['YearsInCurrentRole'].unique().min()
```

```
0
```

```
[26] attrition_data['YearsInCurrentRole'].unique().max()
```

```
18
```

```
[27] attrition_data['YearsSinceLastPromotion'].unique().min()
```

```
0
```

```
[28] attrition_data['YearsSinceLastPromotion'].unique().max()
```

```
15
```

```
[29] attrition_data['YearsWithCurrManager'].unique().min()
```

```
0
```

```
[30] attrition_data['YearsWithCurrManager'].unique().max()
```

```
17
```

As mentioned in the Data Source section earlier, some columns are removed from attrition_data as deemed unnecessary for the analysis. This is to serve the purposes of: (1) using as clean dataset as possible, leaving out the values otherwise serve no purpose, (2) removing the elements that may negatively affect further analysis. For example, the column, "StandardHours", is removed, as all the 1,470 datapoints have 80 hours. The purpose of having a separate DataFrame from attrtion_data is just in case such DataFrame is needed.

```
[31] attrition = attrition_data.drop(['EmployeeCount', 'StandardHours', 'Over1
     attrition.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
       'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'OverTime',
```

```
       'PercentSalaryHike', 'PerformanceRating',
'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

[32]
```
attrition.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1470 entries, 1 to 2068
Data columns (total 31 columns):
Age                         1470 non-null int64
Attrition                   1470 non-null object
BusinessTravel              1470 non-null object
DailyRate                   1470 non-null int64
Department                  1470 non-null object
DistanceFromHome            1470 non-null int64
Education                   1470 non-null int64
EducationField              1470 non-null object
EnvironmentSatisfaction     1470 non-null int64
Gender                      1470 non-null object
HourlyRate                  1470 non-null int64
JobInvolvement              1470 non-null int64
JobLevel                    1470 non-null int64
JobRole                     1470 non-null object
JobSatisfaction             1470 non-null int64
MaritalStatus               1470 non-null object
MonthlyIncome               1470 non-null int64
MonthlyRate                 1470 non-null int64
NumCompaniesWorked          1470 non-null int64
OverTime                    1470 non-null object
PercentSalaryHike           1470 non-null int64
PerformanceRating           1470 non-null int64
RelationshipSatisfaction    1470 non-null int64
StockOptionLevel            1470 non-null int64
TotalWorkingYears           1470 non-null int64
TrainingTimesLastYear       1470 non-null int64
WorkLifeBalance             1470 non-null int64
YearsAtCompany              1470 non-null int64
YearsInCurrentRole          1470 non-null int64
YearsSinceLastPromotion     1470 non-null int64
```

[33]
```
attrition.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Departr
EmployeeNumber					
1	41	Yes	Travel_Rarely	1102	Sales
2	49	No	Travel_Frequently	279	Researcl Develop
4	37	Yes	Travel_Rarely	1373	Researcl Develop

	Age	Attrition	BusinessTravel	DailyRate	Departr
EmployeeNumber					
5	33	No	Travel_Frequently	1392	Researcl Develop
7	27	No	Travel_Rarely	591	Researcl Develop

5 rows × 31 columns

The DataFrame, attrition, has four columns removed from the original DataFrame, attrition_data, leaving out 'EmployeeCount', 'EmployeeNumber', 'StandardHours', 'Over18'.

Below, .describe() is applied to the DataFrame, attrition, except that each column is specified. This way, the result of .describe() will show all the contents.

[34]
```
attrition.describe()
```

	Age	DailyRate	DistanceFromHome	Education	Env
count	1470.000000	1470.000000	1470.000000	1470.000000	147
mean	36.923810	802.485714	9.192517	2.912925	2.72
std	9.135373	403.509100	8.106864	1.024165	1.09
min	18.000000	102.000000	1.000000	1.000000	1.00
25%	30.000000	465.000000	2.000000	2.000000	2.00
50%	36.000000	802.000000	7.000000	3.000000	3.00
75%	43.000000	1157.000000	14.000000	4.000000	4.00
max	60.000000	1499.000000	29.000000	5.000000	4.00

8 rows × 23 columns

Further, the following two separate DataFrame are created: (1) number_data, which consists of only numerica data, and (2) categorical_data, which consists of only categorical data.

[35]
```
number_data = attrition_data[['Age', 'DailyRate', 'DistanceFromHome', 'Ho
```

```
        'JobLevel', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
        'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
        'YearsSinceLastPromotion', 'YearsWithCurrManager']]
number_data
```

	Age	DailyRate	DistanceFromHome	HourlyRate	Jo
EmployeeNumber					
1	41	1102	1	94	2
2	49	279	8	61	2
4	37	1373	2	92	1
5	33	1392	3	56	1
7	27	591	2	40	1
8	32	1005	2	79	1
10	59	1324	3	81	1
11	30	1358	24	67	1
12	38	216	23	44	3
13	36	1299	27	94	2
14	35	809	16	84	1
15	29	153	15	49	2
16	31	670	26	31	1
18	34	1346	19	93	1
19	28	103	24	50	1

[36]
```
categorical_data = attrition_data[['Attrition', 'Gender', 'Education', 'E
                                   'PerformanceRating', 'RelationshipSatisfactio
categorical_data
```

	Attrition	Gender	Education	EnvironmentSatisfaction
EmployeeNumber				
1	Yes	Female	2	2
2	No	Male	1	3
4	Yes	Male	2	4
5	No	Female	4	4
7	No	Male	1	1

	Attrition	Gender	Education	EnvironmentSatisfaction
EmployeeNumber				
8	No	Male	2	4
10	No	Female	3	3
11	No	Male	1	4
12	No	Male	3	4
13	No	Male	3	3
14	No	Male	3	1
15	No	Female	2	4

Applying NumPy's .mean(), .median(), and .mode(), to double-check the results for the selected column, 'MonthlyIncome'.

[37]
```
mean_monthly_income = np.mean(attrition_data['MonthlyIncome'])
median_monthly_income = np.median(attrition_data['MonthlyIncome'])
mode_monthly_income = stats.mode(attrition_data['MonthlyIncome'])

print(mean_monthly_income, median_monthly_income, mode_monthly_income)

#Note the skewness of mean_monthly_income compared to median_monincome
```

```
6502.931292517007 4919.0 ModeResult(mode=array([2342]), count=array([4]))
```

Further, the percentile range (95%) for "MonthlyIncome", the variance, and the standard deviation are calculated to get a further sense for the shape of the dataset:

[38]
```
# Measuring DISPERSION:

print("Monthly Income Percentile Range, Variance, and Standard Deviation
print("95% Percentile Range($): ", np.percentile(attrition_data['MonthlyI
print("Variance($): ", np.var(attrition_data['MonthlyIncome']))
print("Standard Deviation($): ", np.std(attrition_data['MonthlyIncome']))
```

```
Monthly Income Percentile Range, Variance, and Standard Deviation are:
95% Percentile Range($):  2010.175 - 19191.925
Variance($):  22149778.937456165
Standard Deviation($):  4706.355164823004
```

Further notes on the dataset:

- Within in the dataset, there are 588 female and 882 male, the total 1,470 datapoints, with the gender ratio 40% for female, and 60% for male, respectively (see the output for attrition_gender_ratio, below.
- Further, for the Attrition/Yes group, the 37% of those who had attritioin are female while 63% are male.
- For the Attrition/No group, 41% are female while 59% are male.
- This appears as though that for Attrition/No group, there is no apparent gender difference as the gender ratio in this group as the dataset's overal gender ratio is 4(female):6(male).
- This will be explored further below.

[39]
```
attrition_gender_ratio = attrition_data.pivot_table(index='Attrition', co
                                                    fill_value=0)

# calculate ratios
sums = attrition_gender_ratio[['Female', 'Male']].sum(axis=1)
attrition_gender_ratio['FemaleRatio'] = attrition_gender_ratio['Female']
attrition_gender_ratio['MaleRatio'] = attrition_gender_ratio['Male'] / su
attrition_gender_ratio
```

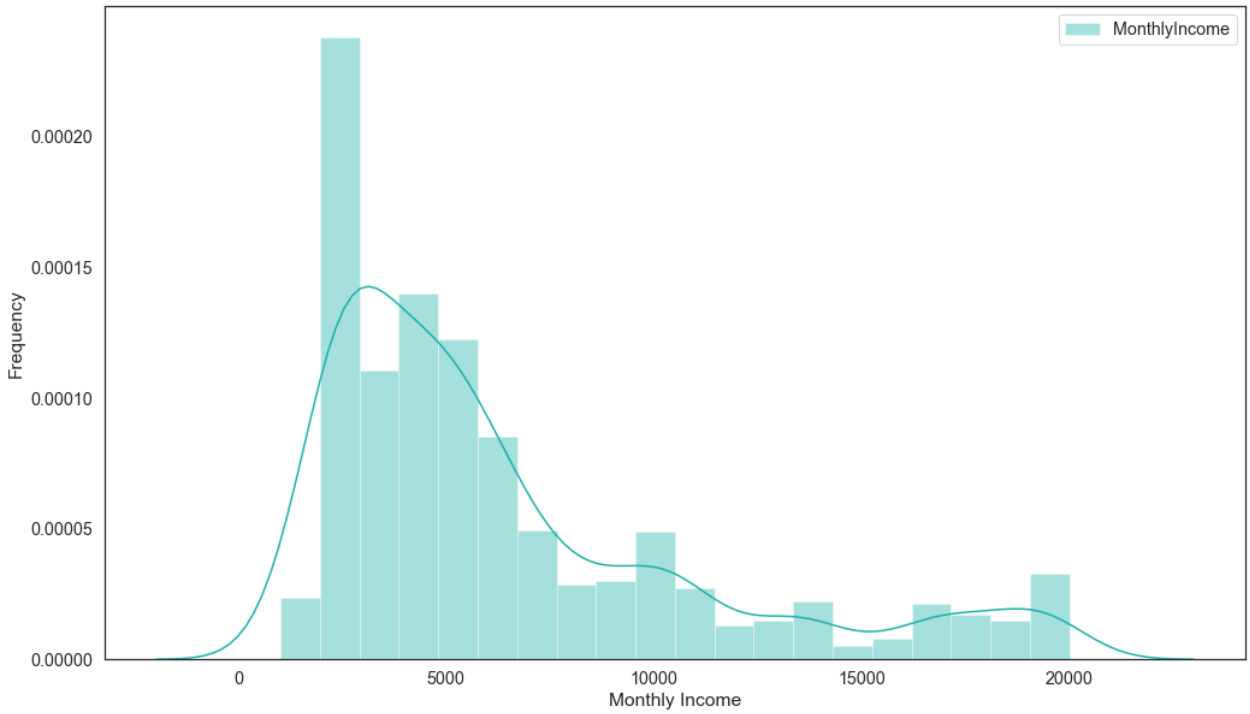
Gender	Female	Male	FemaleRatio	MaleRatio
Attrition				
No	501	732	0.406326	0.593674
Yes	87	150	0.367089	0.632911

Exploratory Question 1: *"How Does Income Income Affect Attrtion?"*:

First, a simple distribution is plotted for MonthlyIncome:

[40]
```
# Plot Female and Male in same plot
fig, axes = plt.subplots(1, 1, figsize=(17, 10))
sns.distplot(attrition_data[['MonthlyIncome']], axlabel=None, label='Mont
plt.legend()
plt.xlabel('Monthly Income')
plt.ylabel('Frequency')
suptitle("Fig. 1: Monthly Income Distribution (Both Genders Combined)")
plt.show()
```

Fig. 1: Monthly Income Distribution (Both Genders Combined)



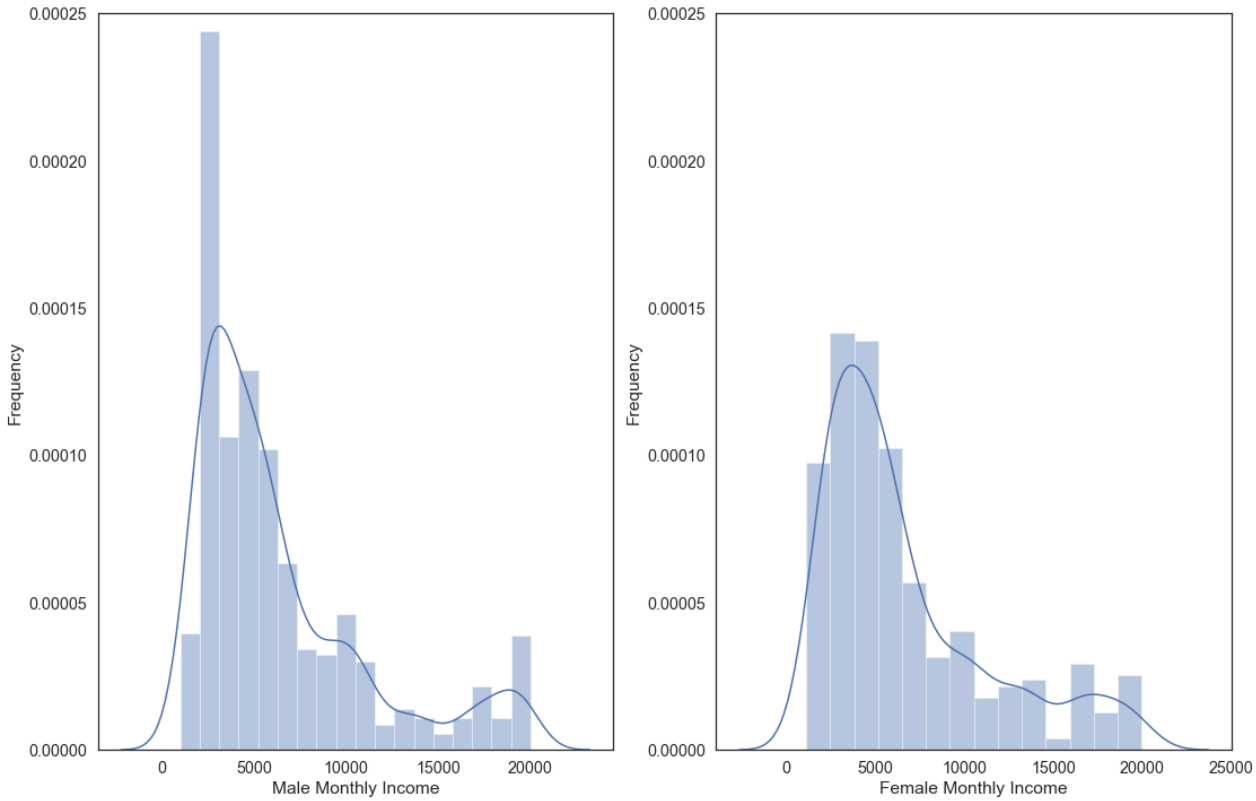
The distribution for Monthly Income is skewed to the right, confirming that the mean for Monthly Income calculated earlier is the result of this skewness as a small portion of the monthly incomes are almost 3-4 times higher than the majority.

Next, MonthlyIncome distributions by Gender will be plotted to see if any apparent gender difference exists:

[41]
```
# Plot Female and Male in same plot
fig, axes = plt.subplots(1, 2, figsize=(18, 12))
attrition_data_M = attrition_data.loc[attrition_data['Gender'] == 'Male']
attrition_data_F = attrition_data.loc[attrition_data['Gender'] == 'Female

sns.distplot(attrition_data_M[['MonthlyIncome']], axlabel=None, label='Ma
sns.distplot(attrition_data_F[['MonthlyIncome']], axlabel=None, label='Fe
axes[0].set_xlabel("Male Monthly Income")
axes[0].set_ylabel("Frequency")
axes[1].set_xlabel("Female Monthly Income")
axes[1].set_ylabel("Frequency")
suptitle("Fig. 2: Monthly Income Distributions by Gender (side-by-side co
ylim = [0, .00025]
axes[0].set_ylim(ylim)
axes[1].set_ylim(ylim)
plt.show()
```

Fig. 2: Monthly Income Distributions by Gender (side-by-side comparison)

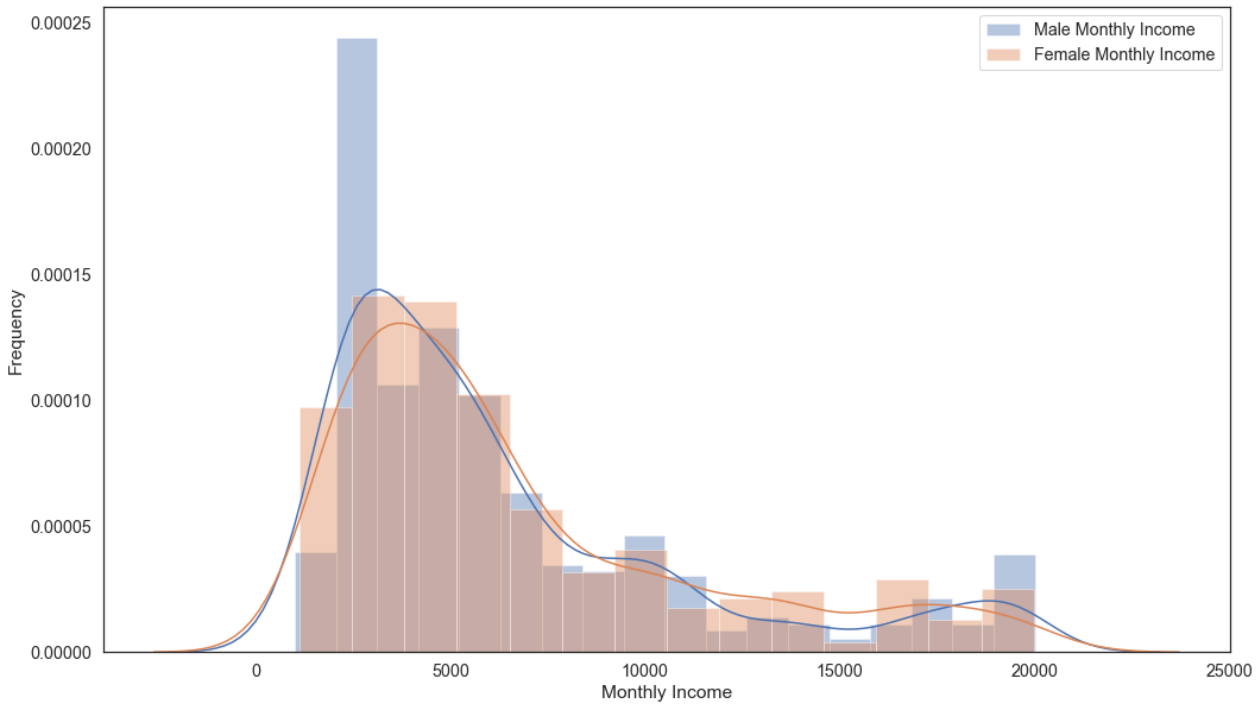


A superimposed version of the above distributions for an easier comparison:

[42]
```
# Plot Female and Male in same plot
fig, axes = plt.subplots(1, 1, figsize=(17, 10))
attrition_data_M = attrition_data.loc[attrition_data['Gender'] == 'Male']
attrition_data_F = attrition_data.loc[attrition_data['Gender'] == 'Female

sns.distplot(attrition_data_M[['MonthlyIncome']], axlabel=None, label='Ma
sns.distplot(attrition_data_F[['MonthlyIncome']], axlabel=None, label='Fe
plt.legend()
plt.xlabel('Monthly Income')
plt.ylabel('Frequency')
suptitle("Fig. 3: Monthly Income Distributions by Gender)")
plt.show()
```

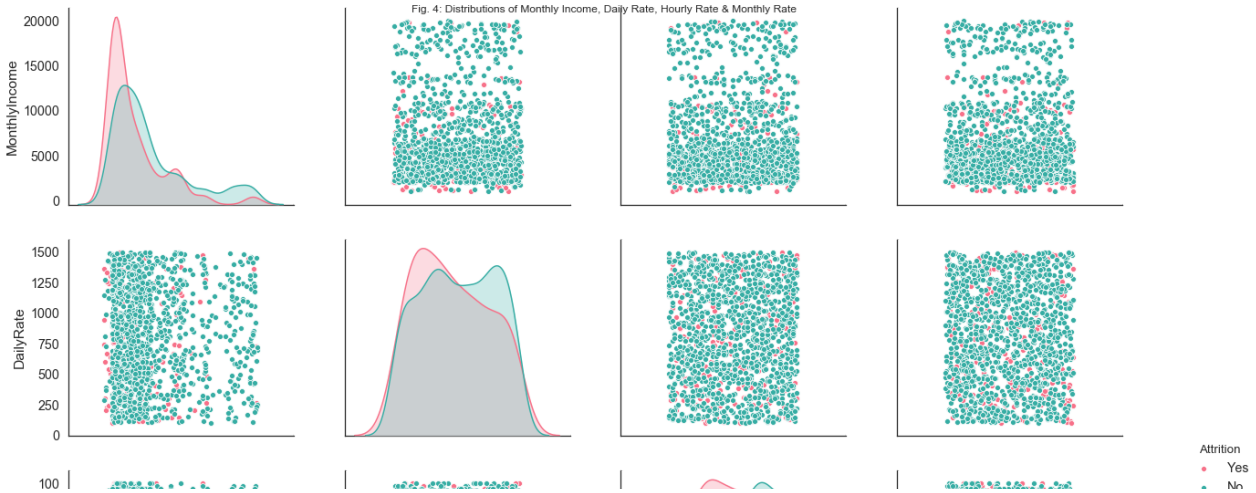
Fig. 3: Monthly Income Distributions by Gender)



In terms of the shapes of the distributions, there is no apparent difference in distributions between the genders.
Next, all the income related numeric data -MonthlyIncome, DailyRate, HourlyRate, and MonthlyRate are plotted for both gender combined to see their respective distributions:

[43]
```
g = sns.pairplot(attrition_data, x_vars=["MonthlyIncome", "DailyRate", "H
                 y_vars=["MonthlyIncome", "DailyRate", "HourlyRate", "Mon
                 palette="husl")
g.fig.set_figheight(15)
g.fig.set_figwidth(20)
suptitle("Fig. 4: Distributions of Monthly Income, Daily Rate, Hourly Rat
```

```
Text(0.5, 0.98, 'Fig. 4: Distributions of Monthly Income, Daily Rate,
Hourly Rate & Monthly Rate')
```



By default, this function will create a grid of Axes such that each variable in data will be shared in the y-axis across a single row and in the x-axis across a single column. The diagonal Axes are treated differently, drawing a plot to show the univariate distribution of the data for the variable in that column.

These pair plot distributions indicate that Monthly Income and Daily Rate indicate in their univariate distributions that the lower the income, higher the attrition will be. A similar is true for HourlyRate and MonthlyRate though not as visibly striking as MontlyIncome and DailyRate.

To make the visual analysis easier, the MonthlyIncome values are convered into ranges (brackets), in which each MonthlyIncome values are categorized into:

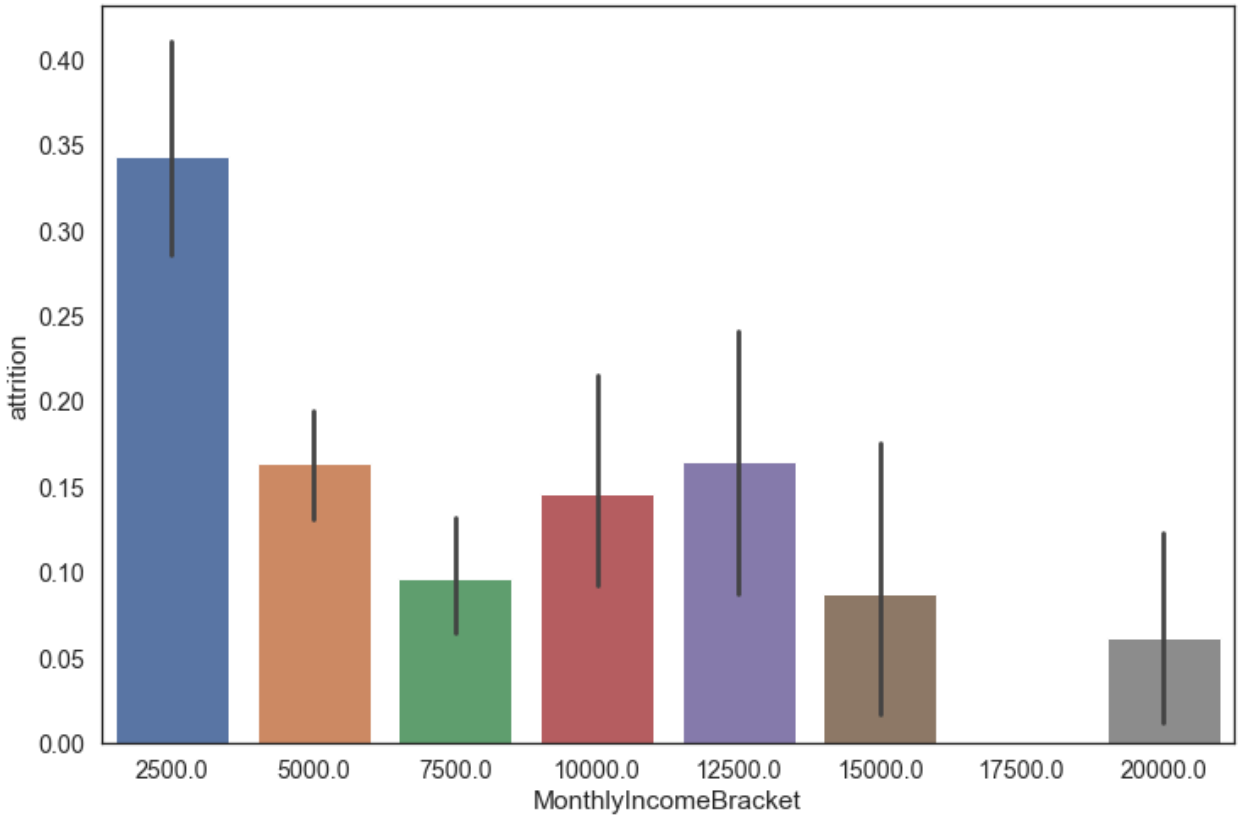
[44]
```
step = 2500
for start in range(0, attrition_data['MonthlyIncome'].max(), step):
  # 0 2500 5000
  rows = (attrition_data['MonthlyIncome'] >= start) & (attrition_data['Mo
  attrition_data.loc[rows, 'MonthlyIncomeBracket'] = start + step
attrition_data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Departr
EmployeeNumber					
1	41	Yes	Travel_Rarely	1102	Sales
2	49	No	Travel_Frequently	279	Researcl Develop
4	37	Yes	Travel_Rarely	1373	Researcl Develop
5	33	No	Travel_Frequently	1392	Researcl Develop
7	27	No	Travel_Rarely	591	Researcl Develop

5 rows × 35 columns

```
[45]  fig, ax = plt.subplots(1, figsize=(12, 8))
      attrition_data['attrition'] = attrition_data['Attrition'].replace({'Yes':
      sns.barplot(x='MonthlyIncomeBracket', y='attrition', data=attrition_data)
      suptitle("Fig. 5: Attrition by Monthly Income Brackets ")
      plt.show()
```

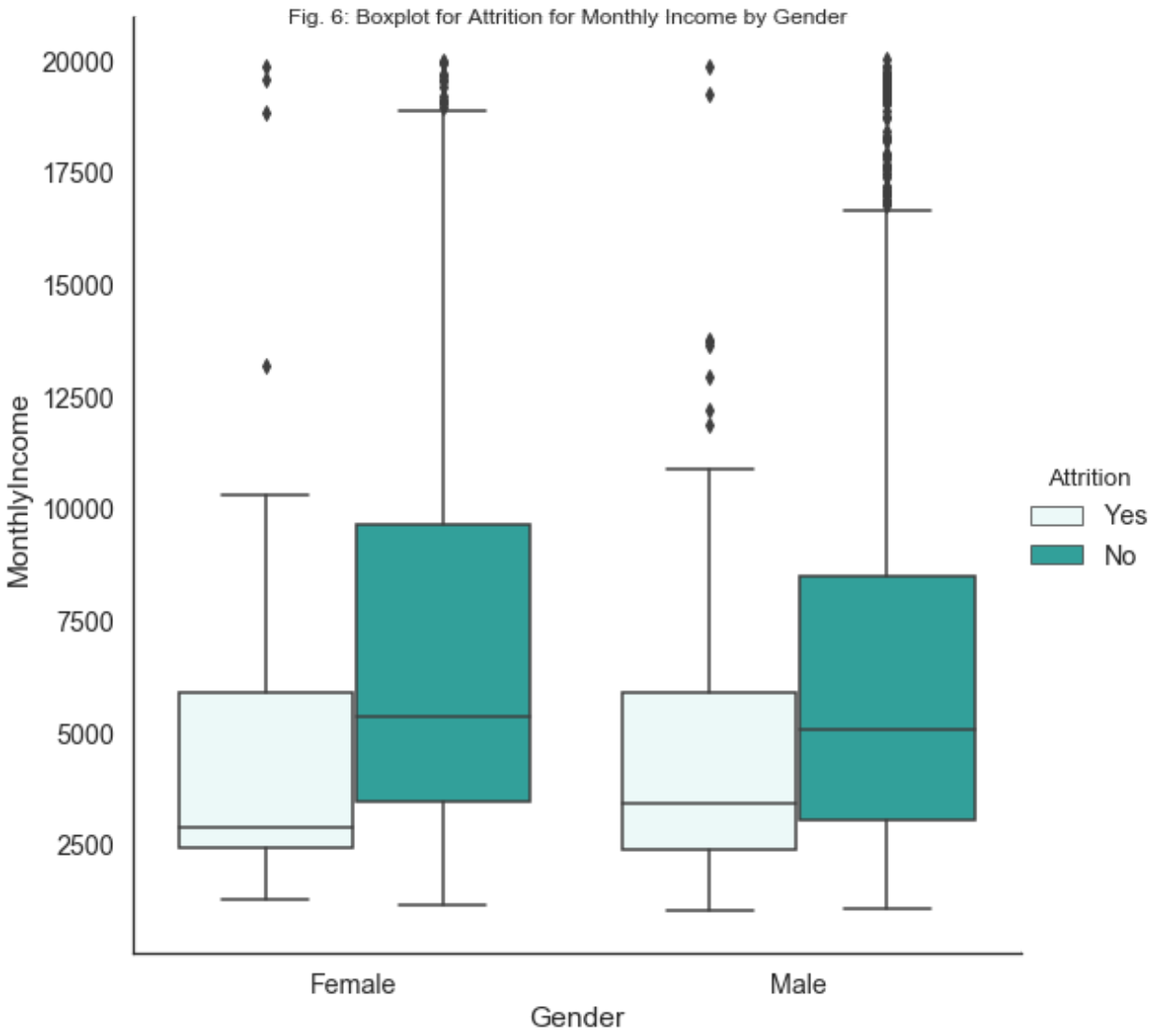
Fig. 5: Attrition by Monthly Income Brackets



As seen in the above barplot, the monthly income group with the highest attrition is the lowest monthly income group (2500 dollars) followed by the next lowest group (5000 dollars). This plot appears to show that the lower the income, higher the attrition is likely. For further contexts, MonthlyIncome, HourlyRate, DailyRate, and MonthlyRate are plotted against Attrition.

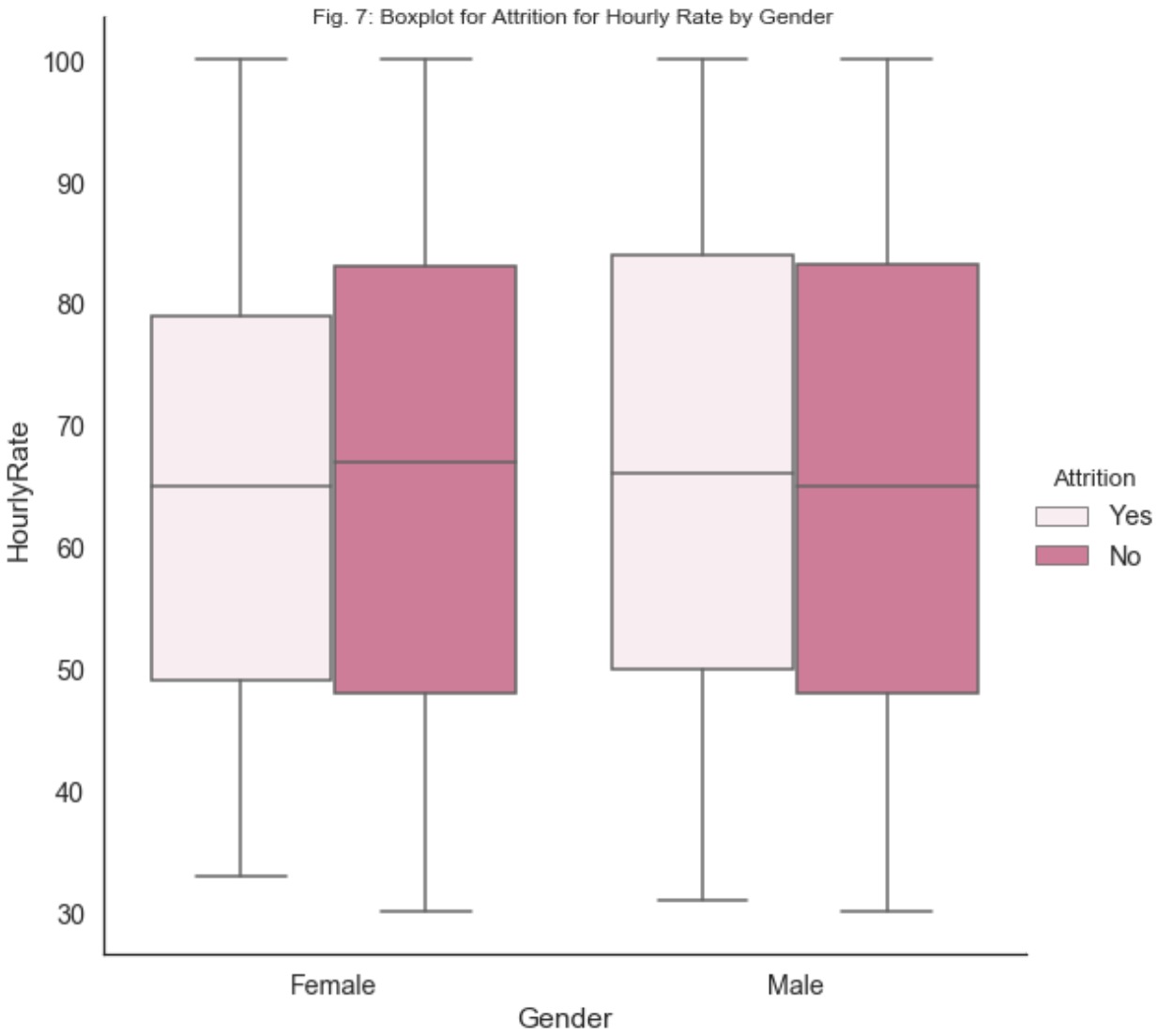
(1) Visualizing the relationship between MonthlyIncome and Attrition:

```
[46]  #fig,axes = plt.subplots(1, figsize=(20,8))
      g = sns.catplot(x='Gender', y='MonthlyIncome', hue='Attrition', data=attr
                      aspect=1, color="lightseagreen")
      suptitle("Fig. 6: Boxplot for Attrition for Monthly Income by Gender ")
      plt.show()
```



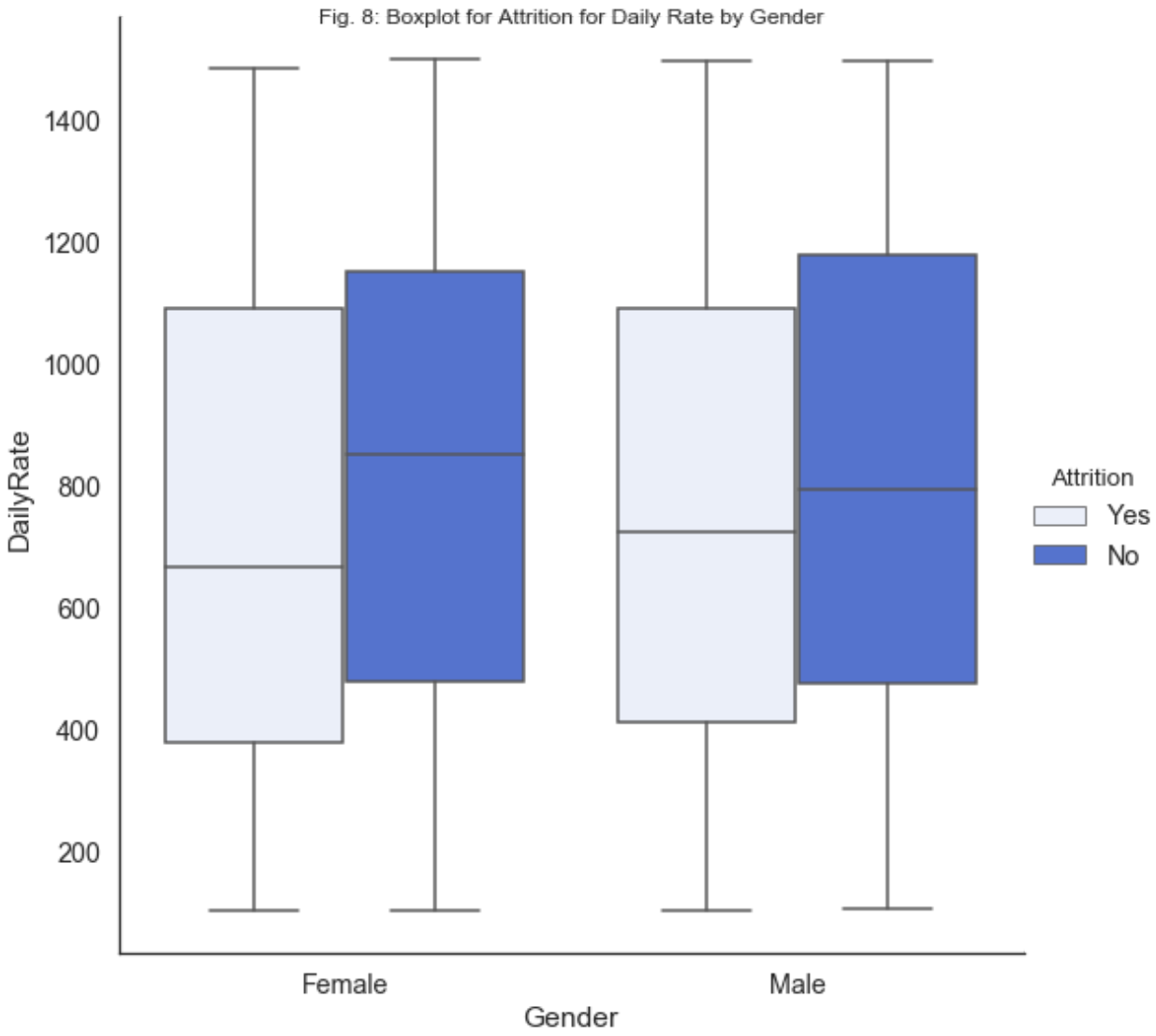
(2) Visualizing the relationship between HourlyRate and Attrition:

```
[47]  g = sns.catplot(x='Gender', y='HourlyRate', hue='Attrition', data=attriti
                      aspect=1, color='palevioletred')
      suptitle("Fig. 7: Boxplot for Attrition for Hourly Rate by Gender ")
      plt.show()
```



(3) Visualizing the relationship between DailyRate and Attrition:

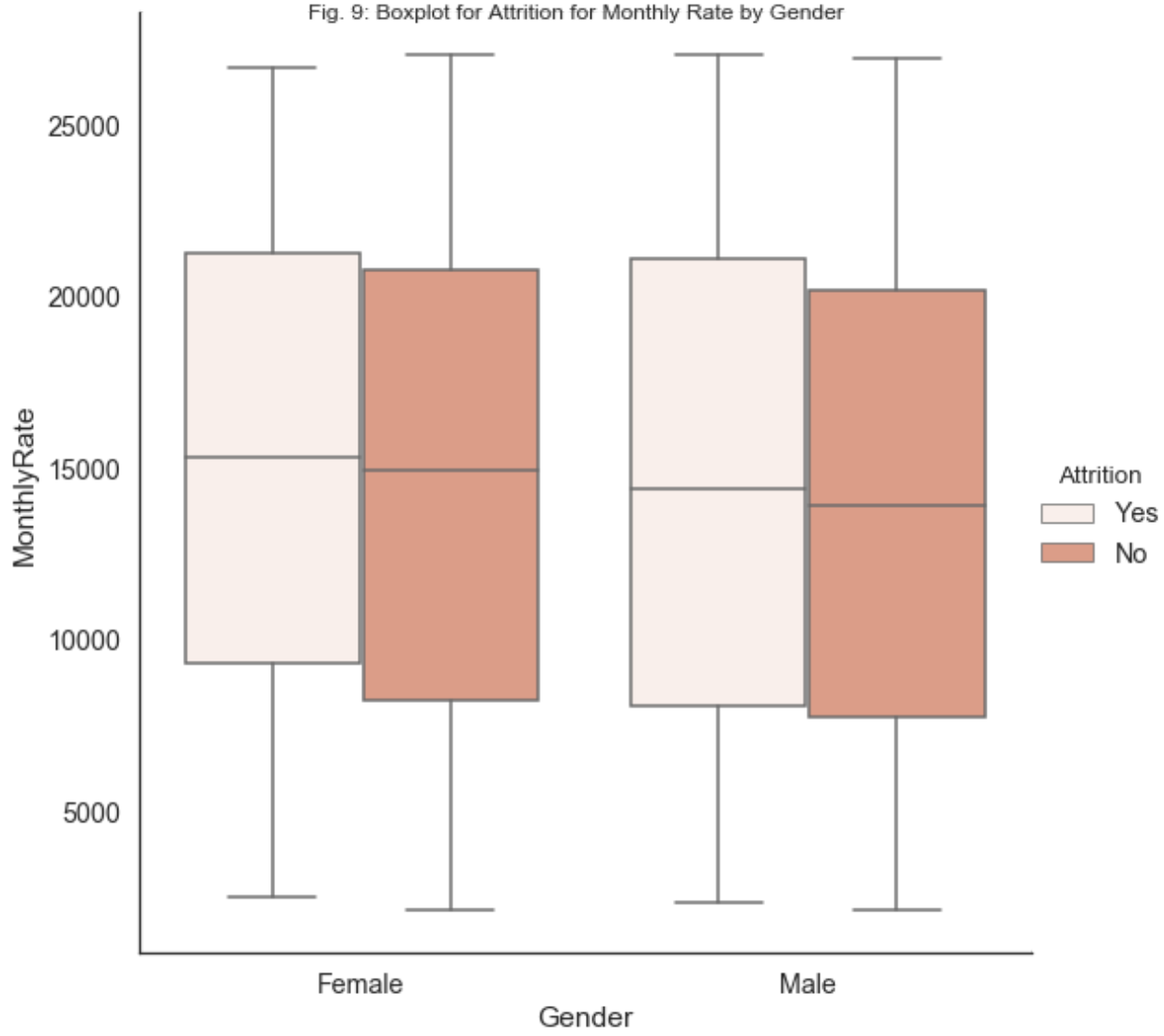
```
[48]  g = sns.catplot(x='Gender', y='DailyRate', hue='Attrition', data=attritio
                      aspect=1, color='royalblue')
      suptitle("Fig. 8: Boxplot for Attrition for Daily Rate by Gender ")
      plt.show()
```



(4) Visualizing the relationship between MonthlyRate and Attrition:

```
[49]  g = sns.catplot(x='Gender', y='MonthlyRate', hue='Attrition', data=attrit
                      kind='box', height=8, aspect=1, color='darksalmon')
      suptitle("Fig. 9: Boxplot for Attrition for Monthly Rate by Gender ")
      plt.show()
```

Fig. 9: Boxplot for Attrition for Monthly Rate by Gender



Observations from the boxplots (1) to (4):

(1) Based on the boxplots above, MonthlyIncome distributions most strikingly show (see the 50-percentile mark) that the individuals with Attrition/Yes tend to have lower incomes in both genders.
The means for both genders shows that the incomes for the 50 percentile are approximately 2,500 dollars for Attrition/Yes group while it is about 5,000 dollars for Attrition/No group. Also notable is how 50% of the income distributions for Attrition/Yes groups in both genders (the boxes) are much shorter than that of its Attrition/No groups. For the female Attrition/No group, the 50% of the distribution lies somewhere between about 3200 dollars to 9600 dollars. As for the male Attrition/No group, the 50% of the distribution lies somewhere between about 3500 dollars to 8500 dollars. This observation is interesting in that just by looking at the 50% distributions alone, and it appears as though female are earning more. However, it is important to note that the male Attrition/No group has more outliers.

(2) The distributions for DailyRate also show that individuals with Attrition/Yes tend to have lower incomes in both genders. The means for the HourlyRate discrepancy for the female shows that the incomes for the 50 percentile are approximately 660 dollars for Attrition/Yes group while it is about 830 dollars for Attrition/No group. The same is true for the male counterparts. While the discrepancy is not as wide as the female one, the differences between the means are approximately between 700 dollars to 750 dollars, which can make a significant difference in their paychecks (Note: No outliers exist).

(3) The distributions for HourlyRate indicates that the female Attrition/Yes earn less than the Attrition/No group while the male Attrition/Yes group is earning slightly more than its Attrition/No group.

(4) For the MonthlyRate distributions, the Attrition/No groups for both male and female have slightly higher than that of the ones for the Attrition/Yes groups.

(5) Although not all the distributions show that the lower rates of income/wages always

[50]
```
stats.ttest_ind(attrition_data[attrition_data.Attrition == 'Yes']['Monthl
                attrition_data[attrition_data.Attrition == 'No']['Monthly
```

```
Ttest_indResult(statistic=-6.203935765608938, pvalue=7.14736398535381e-10)
```

Results:

- The result of the t-test performed above, the calculated p-value (pvalue=7.14736398535381e-10) is significantly smaller than 0.05.
- This indicates the strong evidence against the null hypothesis H0=Income makes no difference in attrition.
- Therefore, I can safely reject the null hypothesis, thus conclude that the alternative hypothesis is true, income does make a difference in attrition.

Before concluding the exploration on the question of income and attrition, the following tabular information is created:

[51]
```
attrition_pivot = attrition_data.pivot_table(index=['Attrition', 'Monthly
                                values='Age', aggfunc=lambda x: len(x) if len(
attrition_pivot
```

	Gender	Female	Male	All
Attrition	**MonthlyIncomeBracket**			
No	**2500.0**	53	94	147
	5000.0	175	264	439
	7500.0	117	163	280
	10000.0	40	71	111
	12500.0	34	42	76
	15000.0	30	22	52
	17500.0	28	24	52

	Gender	Female	Male	All
Attrition	**MonthlyIncomeBracket**			

	20000.0	24	52	76
Yes	**2500.0**	25	52	77
	5000.0	36	50	86
	7500.0	10	20	30
	10000.0	9	10	19

To put the above numbers in perspective, the following pivot table is created to see the gender ratios for each monthly income brackets:

[52]
```
attrition_pivot = attrition_pivot.rename(index={'All': 'Total'}, columns=
attrition_pivot.loc[:, 'Male'] = (attrition_pivot.loc[:, 'Male'] / attrit
attrition_pivot.loc[:, 'Female'] = (attrition_pivot.loc[:, 'Female'] / at
attrition_pivot.loc[:, 'total_prcnt'] = (attrition_pivot.loc[:, 'Total']
attrition_pivot
```

	Gender	**Female**	**Male**	**Total**	**total_prc**
Attrition	**MonthlyIncomeBracket**				
No	**2500.0**	36.054422	63.945578	147	100.0
	5000.0	39.863326	60.136674	439	100.0
	7500.0	41.785714	58.214286	280	100.0
	10000.0	36.036036	63.963964	111	100.0
	12500.0	44.736842	55.263158	76	100.0
	15000.0	57.692308	42.307692	52	100.0
	17500.0	53.846154	46.153846	52	100.0
	20000.0	31.578947	68.421053	76	100.0
Yes	**2500.0**	32.467532	67.532468	77	100.0
	5000.0	41.860465	58.139535	86	100.0
	7500.0	33.333333	66.666667	30	100.0
	10000.0	47.368421	52.631579	19	100.0
	12500.0	20.000000	80.000000	15	100.0
	15000.0	20.000000	80.000000	5	100.0
	20000.0	60.000000	40.000000	5	100.0

Observations:
It is interesting to see that some of the income groups have a similar ratio to the overall dataset ratio (female:4, male:6).
For example, looking at the groups for Attrition/No, the gender ratio (percentage) for the female is anywhere approximately between 36% to 45% for the income groups between 2500 and 12500, staying within the 10% range of the 40%.
The 15000 dollar group for female Attrition/No is almost 58%, the highest percentage, for the female Attrition/No group. Together with the 17500 dollar group, female in this income category show a higher percentage than that of the males through the rate dramatically decreases for 20000 group.

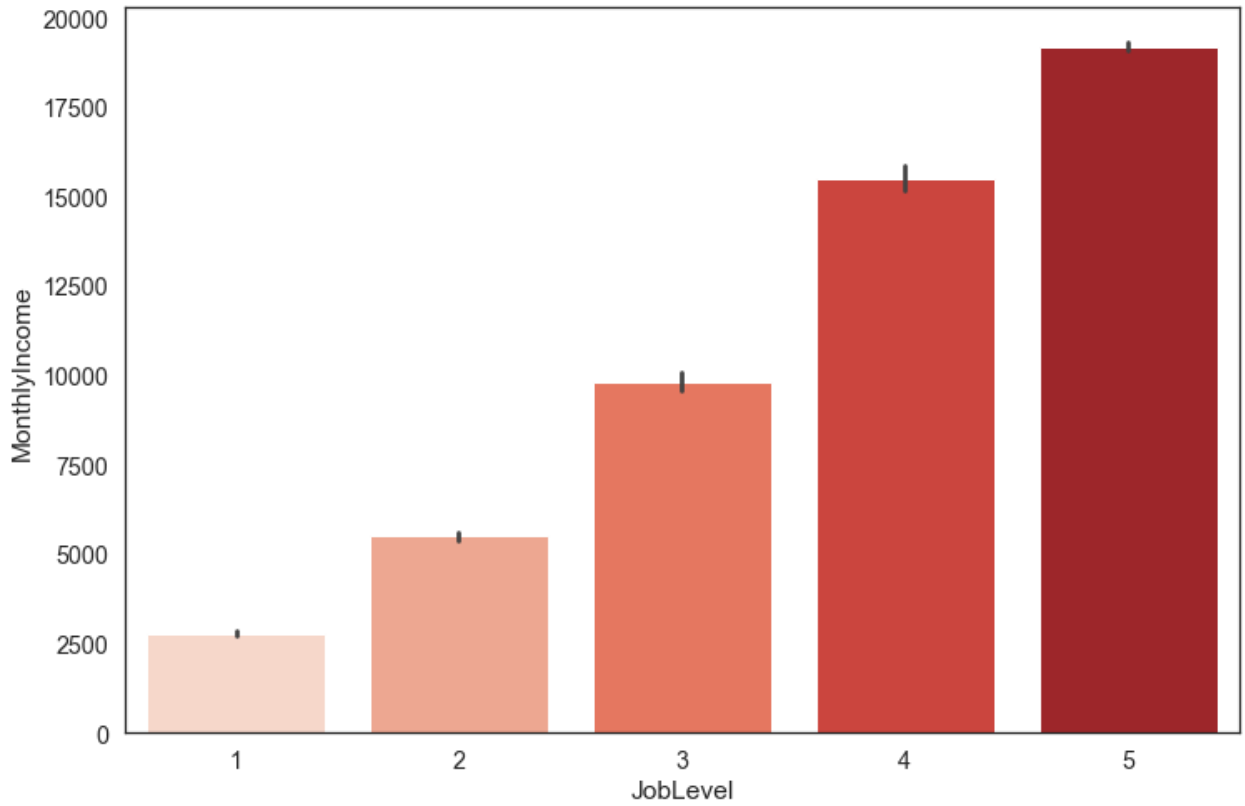
Looking at the Attrition/Yes group, the lower income groups do not seem to have anything striking in terms of gender differences, compared to the Attrition/No group. However, looking at the income groups 12500, 15000, for the Attrition/Yes, the gender ratio is 2:8, meaning that only 20% of female are in these categories (only one female in each of these two groups). Even though the 20000 income group indicates the 60% of the female is in this category, the row total within this category verifies that 3 out of 5 individuals, who are in this income category, are female, which does not appear to be significant. Overall, it seems more notable to see the gender ratio differences in 12500 and 15000 dollars income categories for Attrition/Yes as these may be an indication that female tend to reach the ceiling of highest earning potentials compared to male whatever the case may be.

Next, the topic of the Job Role/Job Level and Attrition is investigated.

Exploratory Question: ***"Can Job Level and/or Job Role affect Attrition?"***:

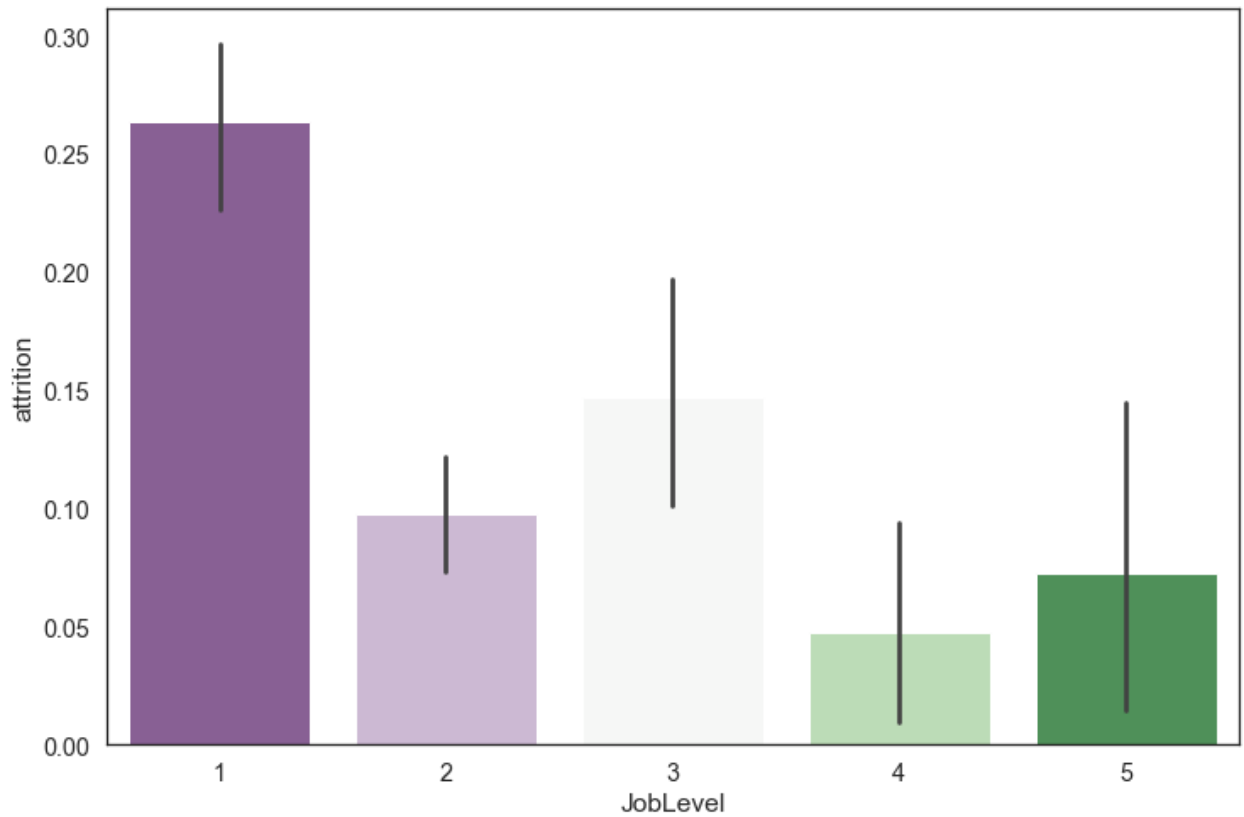
Notes on the relevant columns: This topic involves categorical data mainly, JobLevel and JobRole.
As mentioned earlier, JobLevel consists of 1 to 5 categories while JobRole consists of 9 job titles, (e.g.) Sales Representative, etc. There is no clear group or boundaries between the JobLevel and JobRoles. For instance, JobLevel 1 includes the JobRoles such as Sales Representative, Laboratory Technicians, Research Scientist, etc., while JobLevel 2 can consist of all the said JobRoles. For example, JobRole, "Manager" appears in JobLevel 3, 4, and 5. That said, it is important to note that the lower number the JobLevel is, the income is smaller as shown in the below plot:

[53]
```
fig, ax = plt.subplots(1, figsize=(12, 8))
sns.barplot(x='JobLevel', y='MonthlyIncome', data=attrition_data, palette
plt.show()
```



First, a simple bar plot is created against Attrtion to see if a general trend exists.

[54]
```
fig, ax = plt.subplots(1, figsize=(12, 8))
sns.barplot(x='JobLevel', y='attrition', data=attrition_data, palette='PR
plt.show()
```



Although there is no clean and clear trend is seen, the JobLevel 1 indicates the group's vulnerability to job attrition. The figure for this group, 0.26-0.27 is much higher than the ones for JobLevel 4 and 5 ranging from approximately 0.04 to 0.075. This observation is further confirmed by creating the following crosstab showing the percentages for each group.

[55]
```
joblevel = pd.crosstab(attrition_data['Attrition'], attrition_data['JobLe
joblevel * 100
```

JobLevel	1	2	3	4	5	All
Attrition						
No	27.210884	32.789116	12.653061	6.870748	4.353741	83.87
Yes	9.727891	3.537415	2.176871	0.340136	0.340136	16.12
All	36.938776	36.326531	14.829932	7.210884	4.693878	100.0

The crosstab above shows that 16% of the dataset indicates job attrition (Attrition/Yes), almost 10% of which belongs to the JobLevel 1. This is nearly 63% of the total Attrition/Yes. This observation leads to the following hypothesis:

H0: The JobLevel does not affect Attrition. H1: The JobLevel does affect Attrition (H0 is not True).

One-Way ANOVA will be performed to test this hypothesis. To do so, a list needs to be created first so that the list variable can be passed onto the inside .f_oneway().

[56]
```
joblevels = []
for i in range(1, 6):
  jl = attrition_data.loc[attrition_data['JobLevel'] == i, 'attrition']
  joblevels.append(jl)
stats.f_oneway(*joblevels)
```

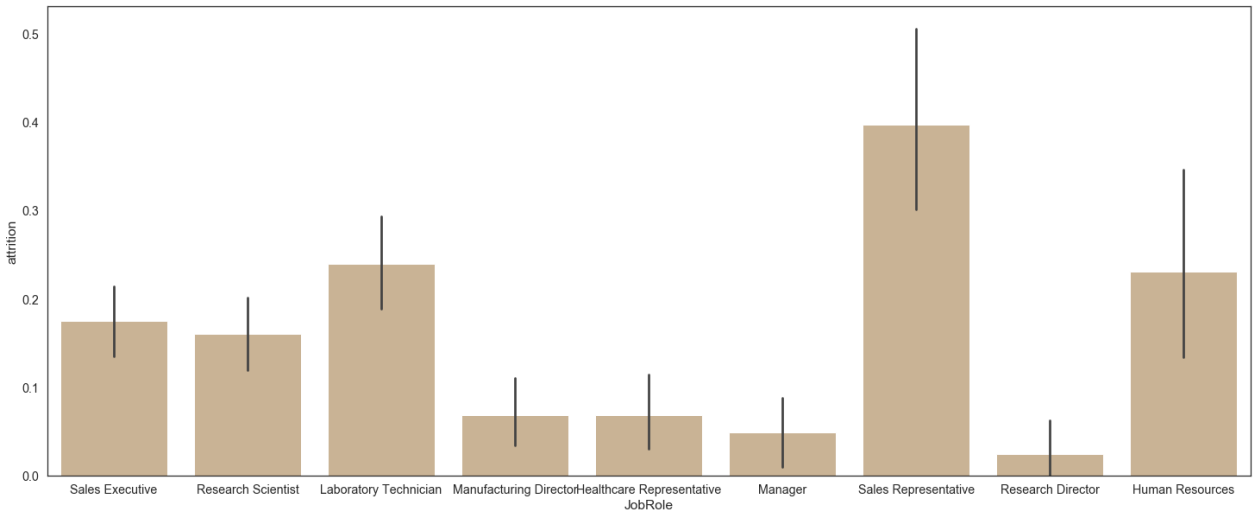
```
F_onewayResult(statistic=19.0084454700361, pvalue=2.975150100310332e-15)
```

The calculated pvalue=2.975150100310332e-15, significantly smaller than 0.05. Therefore, I can safely reject the null hypothesis (H0: The JobLevel does not affect Attrition).

To investigate further, another relevant column, JobRole will be analyzed similarly.

First, a simple barplot is created for JobRoles against Attrition as follows:

[57]
```
fig, ax = plt.subplots(1, figsize=(25, 10))
sns.barplot(x='JobRole', y='attrition', data=attrition_data, color='tan')
plt.show()
```



The bar graph above indicates that the Sales Representative, Laboratory Technician, and Human Resources professionals are more prone to job attrition compared to their counterparts.

The following pivot table and crosstab are created to see the relationship between JobLevel and JobRole.

```
[58]  levelandrole_pivot = attrition_data.pivot_table(index=['JobRole'], column
                                values='Attrition', aggfunc=lambda x: len(x) i
      levelandrole_pivot
```

JobLevel	1	2	3	4	5	All
JobRole						
Healthcare Representative	NaN	78.0	44.0	9.0	NaN	131
Human Resources	33.0	13.0	6.0	NaN	NaN	52
Laboratory Technician	200.0	56.0	3.0	NaN	NaN	259
Manager	NaN	NaN	12.0	47.0	43.0	102
Manufacturing Director	NaN	90.0	45.0	10.0	NaN	145
Research Director	NaN	NaN	28.0	26.0	26.0	80
Research Scientist	234.0	57.0	1.0	NaN	NaN	292
Sales Executive	NaN	233.0	79.0	14.0	NaN	326
Sales Representative	76.0	7.0	NaN	NaN	NaN	83

JobLevel	1	2	3	4	5	All
JobRole						

All	543.0	534.0	218.0	106.0	69.0	1470

[59]
```
jobrole = pd.crosstab(attrition_data['Attrition'], attrition_data['JobRol
                      normalize=True)
jobrole * 100
```

JobRole	Healthcare Representative	Human Resources	Laboratory Technician	Manager	Manufactu Director
Attrition					
No	8.299320	2.721088	13.401361	6.598639	9.183673
Yes	0.612245	0.816327	4.217687	0.340136	0.680272
All	8.911565	3.537415	17.619048	6.938776	9.863946

As the table above indicates, the JobRoles such as Sales Representative, Laboratory Technician, and Human Resources professionals occupy the lower JobLevel, mostly in 1, followed by 2, and very few in 3. As seen earlier, the JobLevel 1 is the group that is particularly high in job attrition.

The following hypothesis is formulated as the result of these observations:

H0: JobRole does not affect Attrition
H1: JobRole does affect Attrition (H0 is not True).

[60]
```
jobroles = []
for j in attrition_data['JobRole'].unique():
  jr = attrition_data.loc[attrition_data['JobRole'] == j, 'attrition']
  jobroles.append(jr)
  # anova needs a list to be paassed onto
print(len(jobroles))
stats.f_oneway(*jobroles)
```

```
9

F_onewayResult(statistic=11.374753732967797, pvalue=9.562555450860023e-16)
```

The calculated pvalue=9.562555450860023e-16, is significantly smaller than 0.05. Therefore, I can safely reject the null hypothesis (H0: The JobRole does not affect Attrition).

For the analyses and results of JobLevel and JobRole taken together, I conclude that JobLevel and JobRole may lead to job attrition.

---

3. Hours of Commitment:

To see if the job attrition can be affected by the total time the individuals may have to spend for in and outside the office associated with work, the following columns are analyzed:
(1) 'DistanceFromHome'
(2) 'MaritalStatus',
(3) 'OverTime',
(4) 'BusinessTravel'

Note: MaritalStatus is included in this analysis based on the possibility that whether the individual is single, married, or divorced, may play a role in the job attrition especially if one must not only travel but also have to have over time, and/or commuting distance is long. Or, it could be the case that the single ones are more likely to have job attritions based on the assumption that they do not have financial obligations to their families and children, allowing them to feel more at ease about resining. Either way, this column seems to be an interesting one to analyze together.

The following code is to categorize commuting distances ranging from 1 to 29 to make analysis and subsequent visualization easier:

[61]
```
step = 5
for start in range(0, attrition_data['DistanceFromHome'].max(), step):
  # 0 2500 5000
  rows = (attrition_data['DistanceFromHome'] >= start) & (attrition_data[
  if start + step > 24:
    attrition_data.loc[rows, 'DistanceRange'] = 25
  else:
    attrition_data.loc[rows, 'DistanceRange'] = start + step
attrition_data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Departr
EmployeeNumber					
1	41	Yes	Travel_Rarely	1102	Sales
2	49	No	Travel_Frequently	279	Researcl Develop
4	37	Yes	Travel_Rarely	1373	Researcl Develop
5	33	No	Travel_Frequently	1392	Researcl Develop
7	27	No	Travel_Rarely	591	Researcl Develop

5 rows × 37 columns

The following is the summary of the selected columns, in which 'DistanceRange' is a new column as the result of the above code to categorize 'DistanceFromHome'.

[62]
```
# Hours of Commitment's relevant colums in a pivot table:
attrition_pivot = attrition_data.pivot_table(index=['OverTime', 'Business
                                             columns=['Gender','Attrition
                                             aggfunc=lambda x: len(x) if
attrition_pivot
```

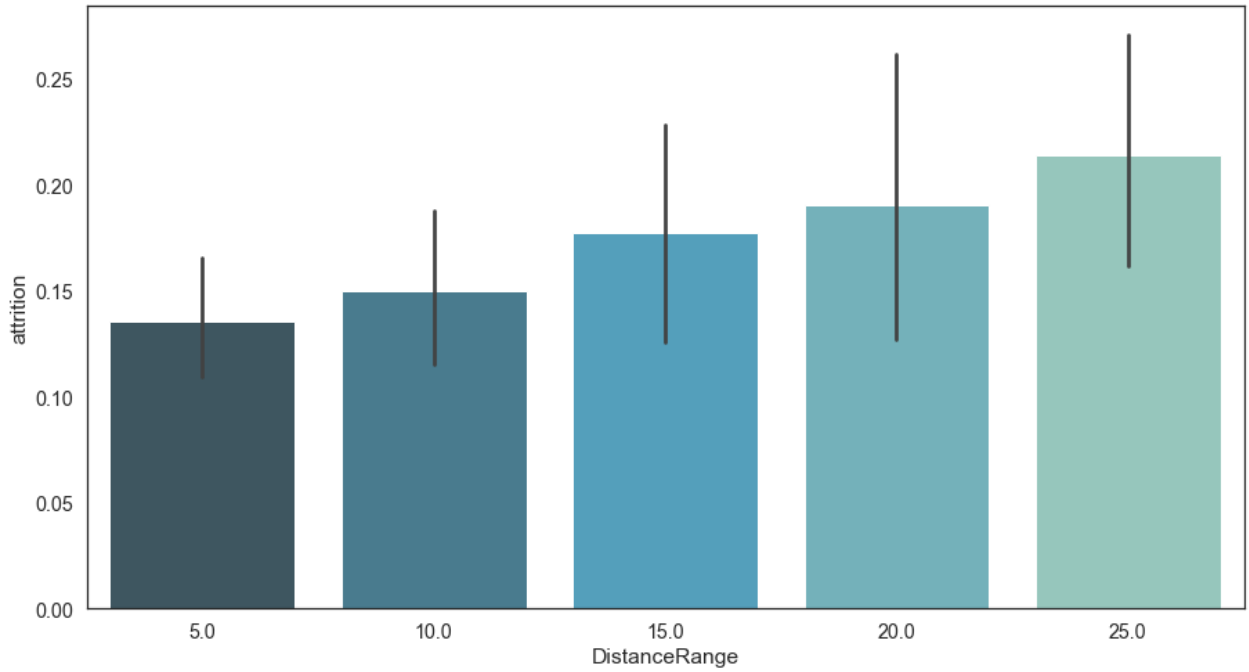
			Gender	Female	
			Attrition	No	Yes
OverTime	BusinessTravel	DistanceRange	MaritalStatus		
No	Non-Travel	5.0	Divorced	2.0	NaN
			Married	8.0	NaN
			Single	4.0	NaN
		10.0	Divorced	1.0	NaN
			Married	4.0	NaN
			Single	2.0	NaN
		15.0	Divorced	1.0	NaN
			Married	2.0	NaN
			Single	1.0	NaN
		20.0	Divorced	1.0	NaN
			Married	NaN	NaN
			Single	NaN	NaN
		25.0	Divorced	2.0	NaN
			Married	4.0	NaN

---

(1) Visualizing the relatipnship between "DistanceFromHome" and "Attrition":

First, plotting DistanceRange and Attrtion in a simple bar graph.

[63]
```
fig, ax = plt.subplots(1, figsize=(15, 8))
sns.barplot(x='DistanceRange', y='attrition', data=attrition_data, palett
plt.show()
```

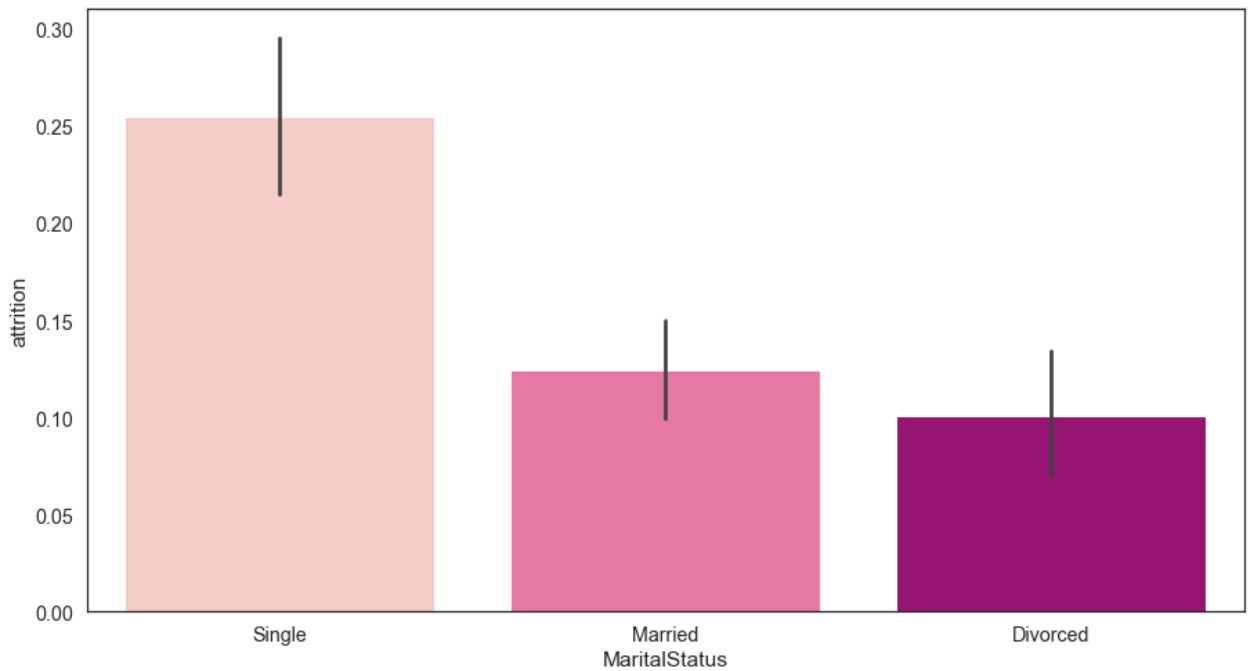


The bargrph above clearly indicates that the longer the distance, the higher the Attrition/Yes is.

Next, for the MaritalStatus and Attrition, bargraphs are plotted.

(2) Visualizing the relatipnship between "MaritalStatus" and "Attrition":

[64]
```
fig, ax = plt.subplots(1, figsize=(15, 8))
sns.barplot(x='MaritalStatus', y='attrition', data=attrition_data, palett
plt.show()
```

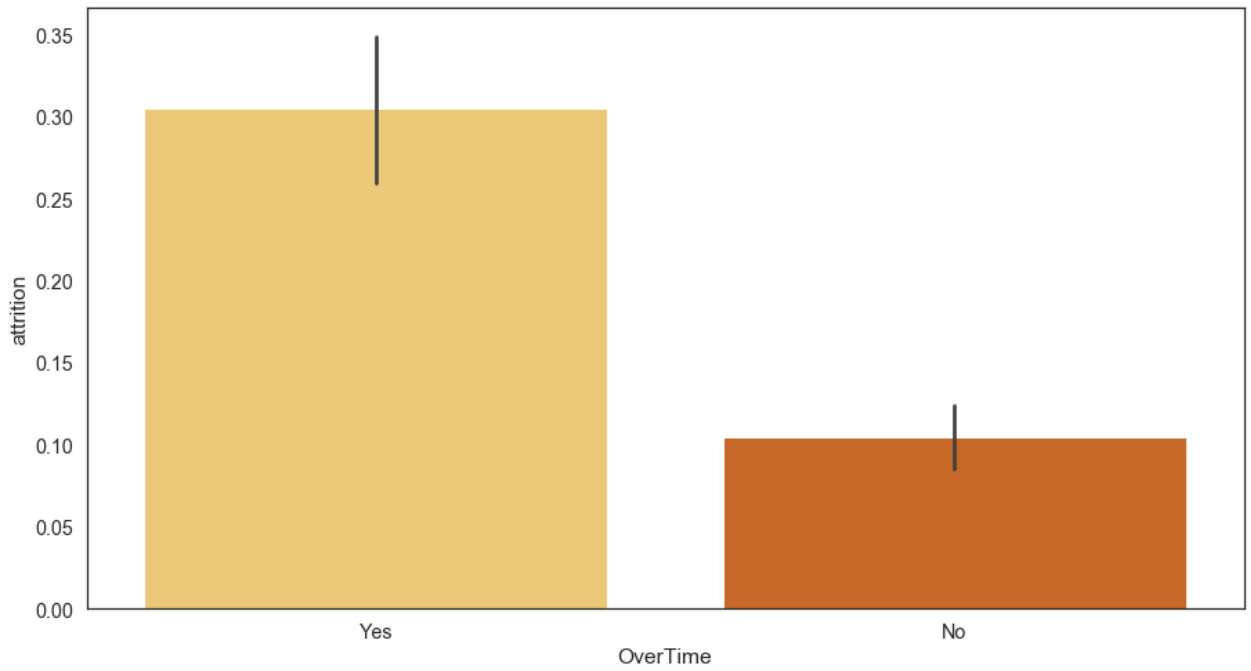


The bar graphs above show that the Single group is more likely to lead to job attrition with the much higher number, little over 0.25, than the ones for Married, and Divorced groups. This may be because, as briefly mentioned earlier, the singles are likely to have fewer obligations to family and children.
Comparing the Single and the Divorced, one may think that why they are strikingly so different although divorced means, they are single in a general categorical sense. However, the difference between the Single and the Divorced may be significant because of the potential financial burdens that the Divorced group may have to their former spouses, and especially to their children. Compared to the Married group, the Divorced group cannot enjoy the tax benefits of filing tax jointly, which is often done by married couples. The Divorced group is also more vulnerable to fewer healthcare benefits compared to its married counterparts. With those high potential financial burdens, the tax, the healthcare, and possibly more, it is after costly to live the life alone while being financially responsible for others. Whatever the true causes may be, this is an interesting graph to see.

---

(3) Visualizing the relatipnship between "OverTime" and "Attrition":

```
[65]  fig, ax = plt.subplots(1, figsize=(15, 8))
      sns.barplot(x='OverTime', y='attrition', data=attrition_data, palette="Yl
      plt.show()
```



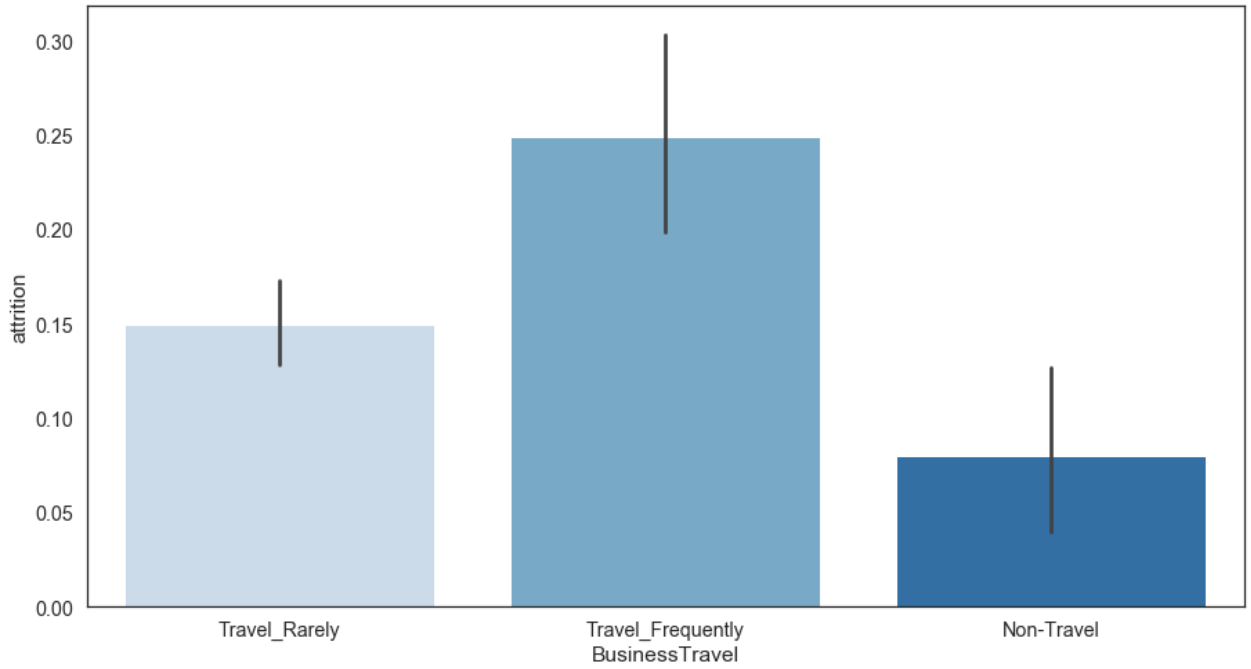
The bar graph above shows a visually striking difference between the Attrition/Yes and Attrition/No group, where the former, the Attrition/Yes group is approximately three times larger than its counterpart, the Attrition/No group.

Based on the p-value, 1.0e-21, there is a strong evidence against the nulll hypothesis, meaning that OverTime does affect Attrition.

---

(4) Visualizing the relatipnship between "BusinessTravel" and "Attrition":

```
[66]  fig, ax = plt.subplots(1, figsize=(15, 8))
      sns.barplot(x='BusinessTravel', y='attrition', data=attrition_data, palet
      plt.show()
```



The bar graph above also shows a trend that the more one travels, the more the individual is likely to have Attrition/Yes.

Finally, One-Way ANOVA will be performed to test the following hypotheses for each column analyzed above:

(1) The DistanceFromHome:
H0: The commuting distance does not affect job attrition.
H1: The commuting distance does affect job attrition (H0 is not True).

(2) MaritalStatus:
H0: The marital status does not affect job attrition.
H1: The marital status distance does affect job attrition (H0 is not True).

(3) OverTime:
H0: The overtime does not affect job attrition.
H1: The overtime does affect job attrition (H0 is not True).

(4) BusinessTravel:
H0: The business travel frequency does not affect job attrition.
H1: The business travel frequency does affect job attrition (H0 is not True).

First, testing for (1) The DistanceFromHome:
H0: The commuting distance does not affect job attrition.
H1: The commuting distance does affect job attrition (H0 is not True).

[67]
```
distance = pd.crosstab(attrition_data['Attrition'], attrition_data['Dista
                       margins=True, normalize=True)
distance * 100
```

DistanceRange	5.0	10.0	15.0	20.0	25.0
Attrition					
No	33.333333	21.564626	9.795918	6.938776	12.244898
Yes	5.238095	3.809524	2.108844	1.632653	3.333333
All	38.571429	25.374150	11.904762	8.571429	15.578231

[68]
```
distances = []
mind = int(attrition_data['DistanceRange'].min())
maxd = int(attrition_data['DistanceRange'].max())
for d in range(mind, maxd + 1, 5):
  dist = attrition_data.loc[attrition_data['DistanceRange'] == d, 'attrit
  distances.append(dist)
  # anova needs a list to be paassed onto
print(len(distances))
stats.f_oneway(*distances)
```

```
5

F_onewayResult(statistic=2.227412086976853, pvalue=0.0639418609871053)
```

The p-value=0.063, meaning that I cannot reject the null hypothesis.

The p-value=0.063, meaning that I cannot reject the null hypothesis.

The above was an unintuitive result for me as I would have guessed that the longer the travel, the more attention is likely. That said, a potential explanation to this is that even the longest commuting distance, the29 in the dataset, the one must travel (assuming it is in miles) may not be considered too much of a commute after all. For the country such as the United States, 29miles are not a long distance if there is highway access. The commuting distance also is correlated to whether the company is located in a city, where there is constant heavy traffic, or located in a rural area with no traffic at all. Either way, it is an interesting result.

Next, testing for (2) MaritalStatus:
H0: The marital status does not affect job attrition.
H1: The marital status distance does affect job attrition (H0 is not True).

[69]
```
mss = []
for val in attrition_data['MaritalStatus'].unique():
  ms = attrition_data.loc[attrition_data['MaritalStatus'] == val, 'attrit
  mss.append(ms)
```

```
print(len(mss))
stats.f_oneway(*mss)
```

```
3
```

```
F_onewayResult(statistic=23.78156546845813, pvalue=6.850067559825624e-11)
```

Based on the p-value=6.85e-11, and I can safely reject the null hypothesis, meaning that H1 is true -The marital status distance does affect job attrition.

For the potential reasons discussed earlier, they may be a variety of causes to why MaritalStatus does affect job attrition. As the Single group is the most likely to have job attrition, this may be related to Age rather than the MaritalStatus in itself.

The next, testing for (3) OverTime:
H0: The overtime does not affect job attrition.
H1: The overtime does affect job attrition (H0 is not True).

[70]
```
ots = []
for val in attrition_data['OverTime'].unique():
  ot = attrition_data.loc[attrition_data['OverTime'] == val, 'attrition']
  ots.append(ot)
  # anova needs a list to be paassed onto
print(len(ots))
stats.f_oneway(*ots)
```

```
2
```

```
F_onewayResult(statistic=94.65645707175152, pvalue=1.0092540336562444e-21)
```

Based on the very low pvalue=1.0092540336562444e-21, I can safely reject the null hypothesis (H0: The overtime does not affect job attrition), meaning that the overtime does affect job attrition. This is an intuitive result, as I know based on my own experience that too long of working hours can lead to 'burn out,' pointing to the workers to resign.

Finally, testing for (4) business travel:
H0: The business travel frequency does not affect job attrition.
H1: The business travel frequency does affect job attrition (H0 is not True).

[71]
```
bts = []
for val in attrition_data['BusinessTravel'].unique():
  bt = attrition_data.loc[attrition_data['BusinessTravel'] == val, 'attri
  bts.append(bt)

print(len(bts))
stats.f_oneway(*bts)
```

```
3
```

```
F_onewayResult(statistic=12.26835294184309, pvalue=5.1998333569549645e-06)
```

Based on the pvalue=5.1998333569549645e-06, I can safely reject the null hypothesis (the business travel frequency does not affect job attrition.), meaning that the business travel frequency does affect job attrition. This is also a straightforward result, again based my own experiences.
Traveling frequently can only enjoyable if they are for leisure. Being stuck at a meeting room outside one's own country, is the same as being held in the office at home.

---

4. Gender and Age Differences in Attrition:

(1) Gender and Attrition:

First, a pivot table by count and crosstab by percentages are created as a summary view. To do so, I first created the below pivot table, which was later added with additional columns with percentages (no_percent, and yes_percent). The total_prcnt column is also created for a complete view.

[72]
```
attrition_by_gender = attrition_data.pivot_table(index='Gender', columns=
                                                 aggfunc='count', margins
attrition_by_gender
# Add precentage so that it is easier to see the ratios --> Do later.
```

Attrition	No	Yes	All
Gender			
Female	501	87	588
Male	732	150	882
All	1233	237	1470

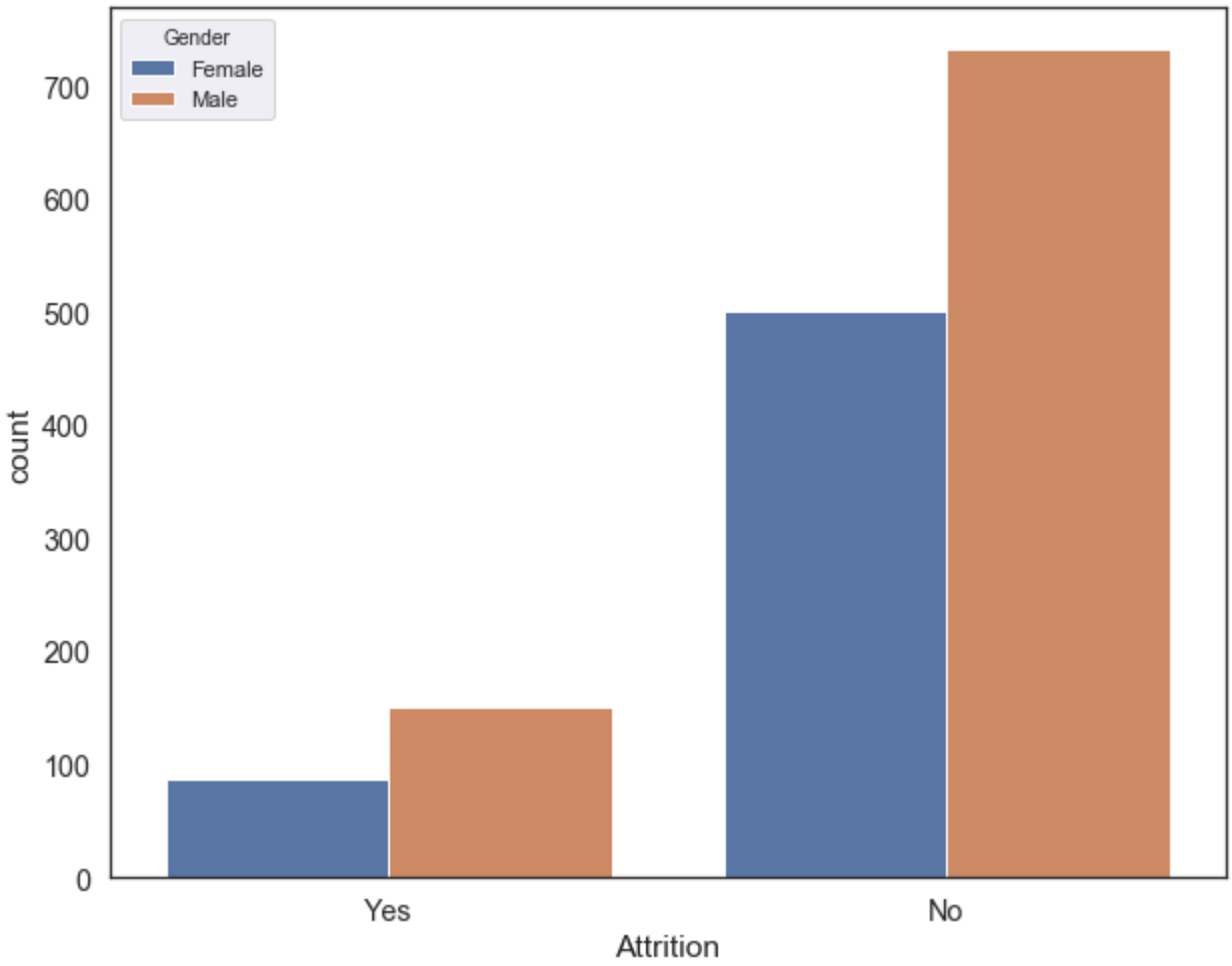
[73]
```
gender = pd.crosstab(attrition_data['Gender'], attrition_data['Attrition'
gender * 100
```

Attrition	No	Yes	All
Gender			
Female	34.081633	5.918367	40.0
Male	49.795918	10.204082	60.0
All	83.877551	16.122449	100.0

To visualize Gender and Attrition, the sns.countplot was used. This is because both the columns are for categorical values: Attrtion: Yes/No, and Gender: Female/Male.

[74]
```
f, ax = plt.subplots(figsize=(10, 8))
sns.set(style="darkgrid")
sns.countplot(x='Attrition', hue="Gender", data=attrition_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x12446d780>
```



First, as mentioned earlier, the ration in Attrition (both genders combined) is Attrition/Yes with 16%, and Attrition/No with 84%.

Taking the results from the crosstab above, in which Looking at the output above, the ratios for the Attrition/Yes and Attrition/No in respective gender are approximately 40% for female and 60% for male.
H0: The two categorical data (Gender and Attrition) 5.9% of Attrition/Yes is female while 10.2 of Attrition/Yes is male. It suggests that the out of 16%, the total Attrition/Yes, about 37% are female while the rest, 63% are male, similar to the proportion of female : male, 4 : 6, included in the dataset, which suggests the following hypothesis:

H0: The two categories (Attrition and Gender) are independent of each other. H1: The two categories (Attrition and Gender) are not independent of each other (H0 is not True).

To test these categorical values, the following Chi-Square test is performed:

[75]
```
# Insert your code here
from scipy.stats import chi2_contingency
chi2, p, dof, ex = chi2_contingency(attrition_by_gender)
print("chi2 = ", chi2)
print("p-val = ", p)
print("degree of freedom = ",dof)
print("Expected:")
pd.DataFrame(ex)

### The code needs to be convered from numbers to string values.
```

```
chi2 =  1.2752163602205182
p-val =  0.8655661914618858
degree of freedom =  4
Expected:
```

	0	1	2
0	493.2	94.8	588.0
1	739.8	142.2	882.0
2	1233.0	237.0	1470.0

Looking at the p-value = 0.8655661914618858, there is strong evidence for the null hypothesis (H0: the two categorical data (Gender and Attrition) are independent of each other. Therefore, I cannot reject the null hypothesis.

This test result indicates that the potential gender difference I thought we might see is a chance, not due to a dependency that exists between the two.

My prior was that there is a gender difference in job attrition as I assumed that the female working population is more vulnerable to it due to various social challenges in and outside work women face. However, after the analysis and the test result, I can deduce that there is no apparent gender difference in attrition because of a self-selecting nature of career options women may be making. For instance, if a woman knows the most of the family obligations fall onto her shoulder (if she has a family with children), she may not choose the type of work that is likely to make her more hectic to the level that she would end up leaving her workplace. She may not want a kind of work that is known to have a lot of overtime, business travels, etc. Perhaps, this result is also an indication that the dataset has high dimensionality, as many of the factors are hard to separate and interdependent.

---

(2) Age and Attrition:

Finally, Age and Attrition will be analyzed. But first, a new colum, AgeRange will be created to categorize the age data in 'Age' column, which ranges between 18 and 60. The ragnes are divided by 5-year interval, except the first age range (18-25).

[76]
```
step = 5
```

```
for start in range(25, attrition_data['Age'].max(), step):
  age_rows = (attrition_data['Age'] >= start) & (attrition_data['Age'] <=
  if start + step > 59:
    attrition_data.loc[age_rows, 'Age'] = 60
  else:
    attrition_data.loc[age_rows, 'AgeRange'] = start + step
attrition_data.loc[attrition_data['Age'] <= 25, 'AgeRange'] = 25

attrition_data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Departr
EmployeeNumber					
1	41	Yes	Travel_Rarely	1102	Sales
2	49	No	Travel_Frequently	279	Researcl Develop
4	37	Yes	Travel_Rarely	1373	Researcl Develop
5	33	No	Travel_Frequently	1392	Researcl Develop
7	27	No	Travel_Rarely	591	Researcl Develop

5 rows × 38 columns

Using the newly created column, 'AgeRange', the following pivot table with percentages is created as an overview.

[77]
```
attrition_by_age = attrition_data.pivot_table(index='AgeRange', columns='
                                              aggfunc='count', margins
```

[78]
```
attrition_by_age = attrition_by_age.rename(index={'All': 'Total'}, column
attrition_by_age.loc[:, 'no_prcnt'] = (attrition_by_age.loc[:, 'No'] / at
attrition_by_age.loc[:, 'yes_prcnt'] = (attrition_by_age.loc[:, 'Yes'] /
attrition_by_age.loc[:, 'total_prcnt'] = (attrition_by_age.loc[:, 'Total'
attrition_by_age
```

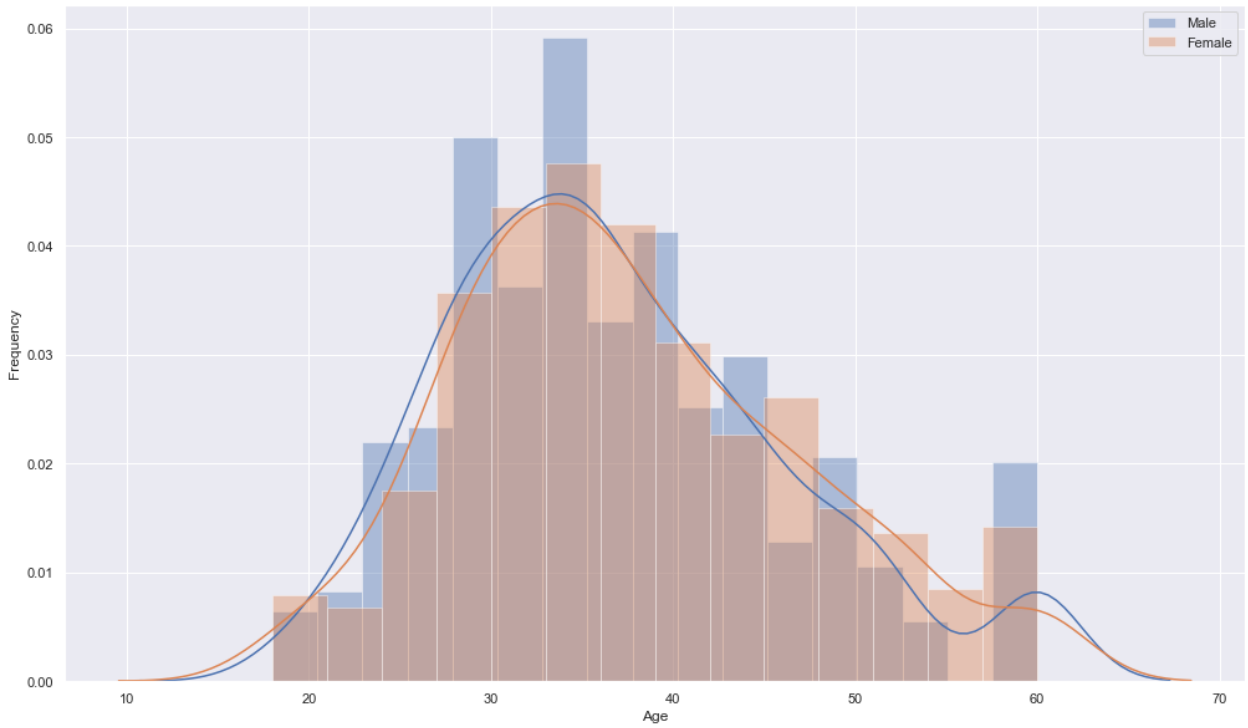
Attrition	No	Yes	Total	no_prcnt	yes_prcnt	total_prcnt
AgeRange	**No**	**Yes**	**Total**	**no_prcnt**	**yes_prcnt**	**total_prcnt**
AgeRange						
25.0	79	44	123	64.227642	35.772358	100.0

30.0	156	47	203	76.847291	23.152709	100.0
35.0	266	59	325	81.846154	18.153846	100.0
40.0	267	30	297	89.898990	10.101010	100.0
45.0	187	21	208	89.903846	10.096154	100.0
50.0	128	13	141	90.780142	9.219858	100.0
55.0	111	15	126	88.095238	11.904762	100.0
Total	1194	229	1423	83.907238	16.092762	100.0

First, a histogram with KDE for age distribution is plotted.

[79]
```
# Plot Female and Male in same plot
fig, axes = plt.subplots(1, 1, figsize=(17, 10))
attrition_data_M = attrition_data.loc[attrition_data['Gender'] == 'Male']
attrition_data_F = attrition_data.loc[attrition_data['Gender'] == 'Female

sns.distplot(attrition_data_M[['Age']], axlabel=None, label='Male')
sns.distplot(attrition_data_F[['Age']], axlabel=None, label='Female')
plt.legend()
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

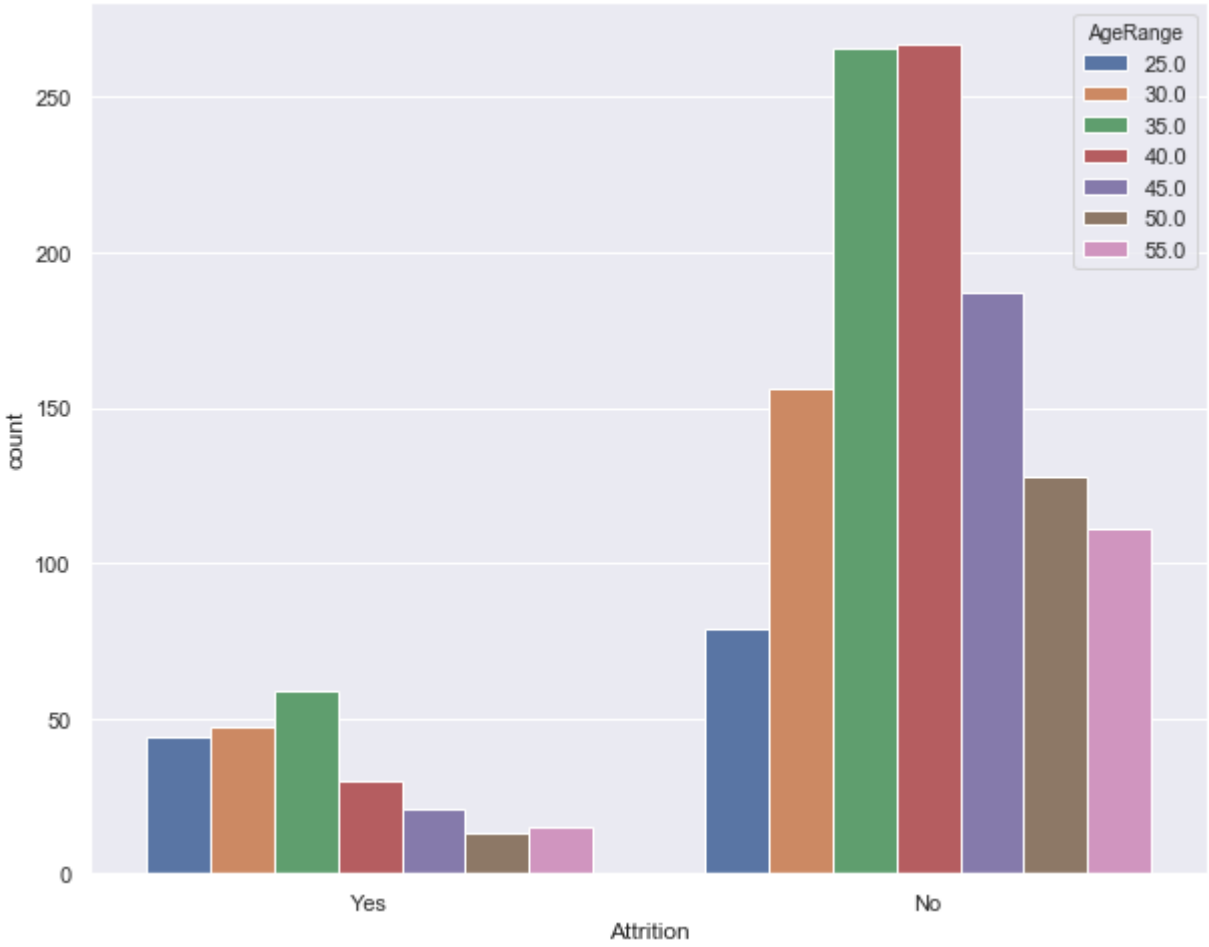


The above age distributions show normal distributions for both genders, where the age range for the highest frequencies occurring are between the 30s and 40s.

The following count plot is plotted for AgeRage and Attrition.

[80]
```
f, ax = plt.subplots(figsize=(10, 8))
sns.set(style="darkgrid")
sns.countplot(x="Attrition", hue="AgeRange", data=attrition_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1249b4588>
```



Though the y-axis is the total count for the each AgeRange, the graph for Attrition/No appears similar to the KDE normal curve for AgeRange plotted earlier. However, for the Attrition/Yes, the younger age groups, the 20s, and the 30s have higher counts than the older age groups, the 40s and above.

The result is curious as an earlier discussion on the marital status, and attrition mentions the higher job attrition for the Single group may be the result of the relationship between the age and erosion.

Based on this observation, I will hypothesize the following:

H0: Age has no effect in attrition.
H1: Age does have an effect in attrition.

To test the hypothesis, the following One-Way ANOVA is performed.

[81]
```
ages = []
mina = int(attrition_data['AgeRange'].min())
maxa = int(attrition_data['AgeRange'].max())
for a in range(mina, maxa + 1, 5):
  age = attrition_data.loc[attrition_data['AgeRange'] == a, 'attrition']
  ages.append(age)
  # anova needs a list to be paassed onto
print(len(ages))
stats.f_oneway(*ages)
```

```
7

F_onewayResult(statistic=11.07747452064328, pvalue=4.226676719467157e-12)
```

Based on the pvalue=4.226676719467157e-12, there is strong evidence against the null hypothesis (H0: Age does not affect attrition). Therefore, I can safely reject the null hypothesis and accept the alternative hypothesis, H1: Age does affect attrition.

This concludes the analysis of the four exploratory questions.

Next, a correlation heat map is created for the numerical data to see interdependencies among numeric data attributes.

Visualizing Correlations for numerical data:

Creating a heatmap to show the correaltions within the number_data:

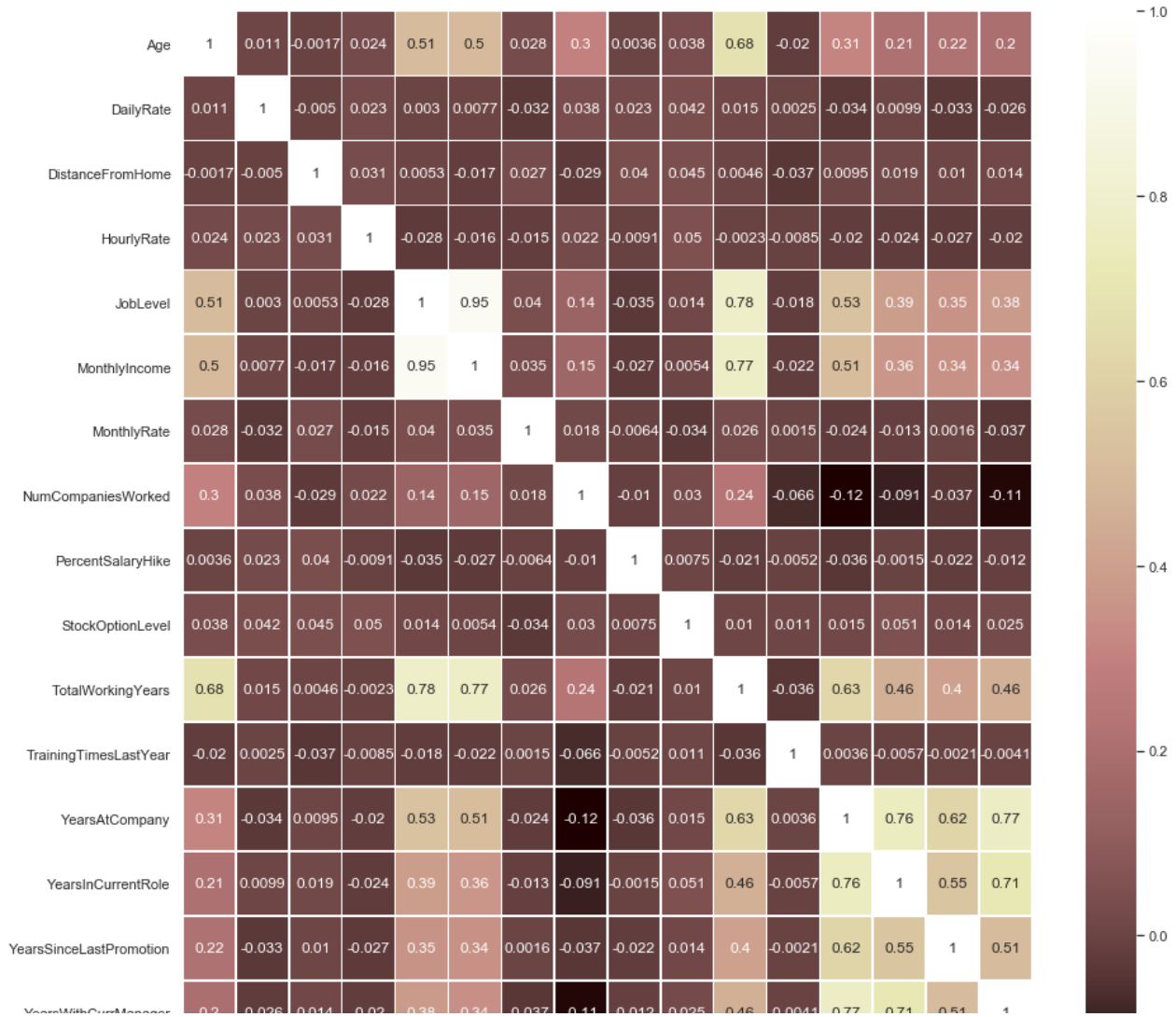
[82]
```
num_data_correlation = number_data.corr()
num_data_correlation
```

	Age	DailyRate	DistanceFromHome	Hourl
Age	1.000000	0.010661	-0.001686	0.024
DailyRate	0.010661	1.000000	-0.004985	0.023
DistanceFromHome	-0.001686	-0.004985	1.000000	0.031
HourlyRate	0.024287	0.023381	0.031131	1.000
JobLevel	0.509604	0.002966	0.005303	-0.027
MonthlyIncome	0.497855	0.007707	-0.017014	-0.015
MonthlyRate	0.028051	-0.032182	0.027473	-0.015
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.022

	Age	DailyRate	DistanceFromHome	Hourl
PercentSalaryHike	0.003634	0.022704	0.040235	-0.009
StockOptionLevel	0.037510	0.042143	0.044872	0.050
TotalWorkingYears	0.680381	0.014515	0.004628	-0.002
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.008
YearsAtCompany	0.311309	-0.034055	0.009508	-0.019
YearsInCurrentRole	0.212901	0.009932	0.018845	-0.024

[83]
```
f, ax = plt.subplots(figsize=(15, 15))
num_data_correlation_map = sns.heatmap(num_data_correlation, annot=True,
num_data_correlation_map
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x124cd2128>
```



The results:
The following is the list of the clumns (attributes) in the order of highest correlations:
(1) JobLevel and MonthlyIncome (0.95)
(2) JobLevel and TotalWorkingYears (0.78)

(3) MonthlyIncome and TotalWorkingYears (0.77)
(4) YearsAtCompany and YearsWithCurrManager (0.77)
(5) YearsAtCompany and YearsInCurrentRole (0.76)
(6) YearsWithCurrManager and YearsInCurrentRole (0.71)

Analysis and interpretations:

- JobLevel is highly correlated to MonthlyIncome and TotalWorkingYears: This result is likely because that the longer you serve, the higher you are likely to be promoted. And higher the position is, the more one is likely to earn as well. This expected result also explains the high correlation in (3) MonthlyIncome and TotalWorkingYears (0.77).

- Some of the "Years" columns are highly correlated with each other. For example, YearsAtCompany is highly correlated to YearsWithCurrentManager and YearsInCurrentRole. These correlations are interesting in that YearsAtCompany can be affected either positively and negatively. For instance, if YearsAtCompany is long, it may also be the case that YearsWithCurrManager if you like working for your manager.

But the opposite can be true, and both YearsAtCompany and YearsWithCurrManager may be short if you do not like working for your manager. A similar case can be made for the relationship between YearsWithCurrManager and YearsInCurrentRole.

The critical thing to note as result of this correlation map is that there appear to be many interrelationships at work, which makes the analysis challenging.

Classifications

Random Forest to predict most important features for 'Attrition':

[84]
```
categorical_data.loc[:, 'Gender'] = categorical_data.replace({'Male': 1,
dropped_attrition = categorical_data.drop('Attrition', axis=1)
dropped_attrition.head()
```

	Gender	Education	EnvironmentSatisfaction	JobInvol
EmployeeNumber				
1	0	2	2	3
2	1	1	3	2
4	1	2	4	2
5	0	4	4	3
7	1	1	1	3

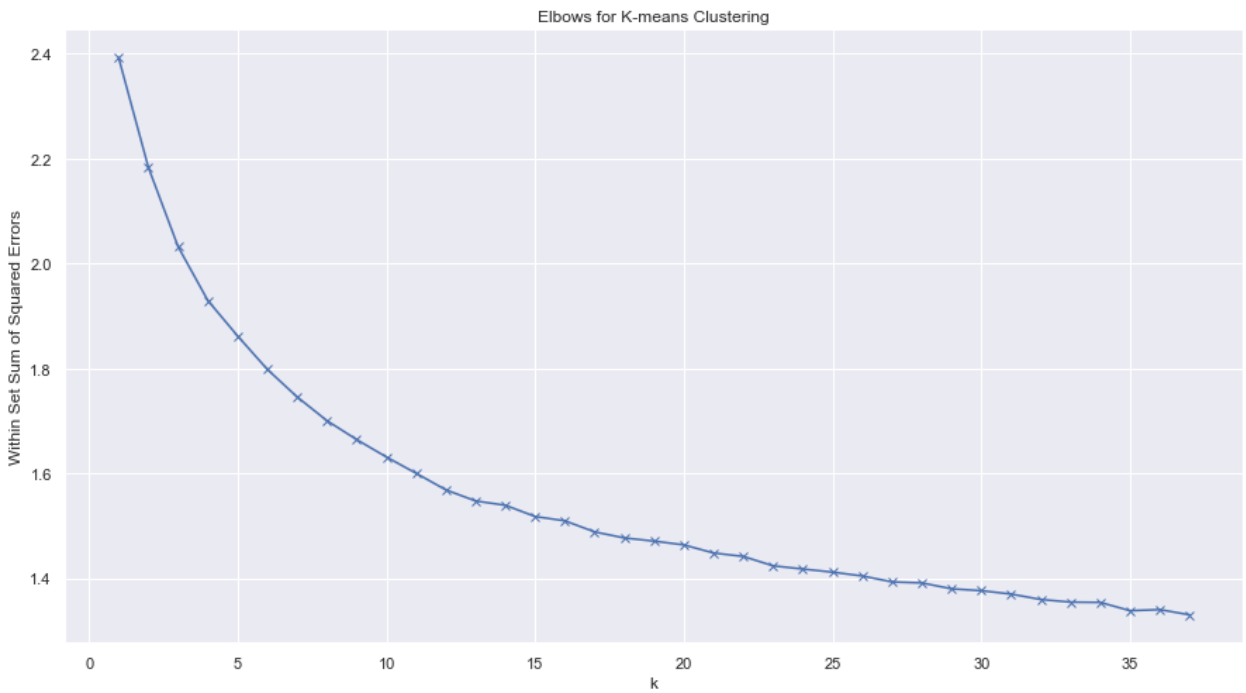
[85]
```
### K-means may be sufficient.
import sklearn as sk
from sklearn import metrics
```

```
# Insert your code here
#sklearn.metrics.silhouette_score(X, labels, metric='euclidean'
#, sample_size=None, random_state=None, **kwds)


from sklearn.cluster import KMeans
from sklearn import metrics
from scipy.spatial.distance import cdist
import numpy as np
import matplotlib.pyplot as plt


#K-means: Derermine K:
distortions = []
K = range(1,38)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(dropped_attrition)
    distortions.append(sum(np.min(cdist(dropped_attrition, kmeanModel.clu

# Elbow:
fig, axix = plt.subplots(figsize=(15, 8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Within Set Sum of Squared Errors')
plt.title('Elbows for K-means Clustering')
plt.show()
```



The above elbow plot suggests the 5 cluster, which will be used below. The resulting clusters with respective assigned cluster numbers are added in kmeans_2 column below.

[86]
```
dropped_attrition = categorical_data.drop('Attrition', axis=1)
dropped_attrition.head()
```

	Gender	Education	EnvironmentSatisfaction	JobInvol
EmployeeNumber				
1	0	2	2	3
2	1	1	3	2
4	1	2	4	2
5	0	4	4	3
7	1	1	1	3

[87]
```
categorical_data.loc[:, 'Attrition'] = categorical_data.replace({'Yes': 1
```

[88]
```
#K-means: Derermine K:
kmeanModel = KMeans(n_clusters=5)
kmeanModel.fit(categorical_data)
categorical_data.loc[:, 'kmeans_2'] = kmeanModel.predict(categorical_data
categorical_data.loc[:, 'Attrition'] = categorical_data['Attrition']
categorical_data.head(20)
```

	Attrition	Gender	Education	EnvironmentSatisfaction
EmployeeNumber				
1	1	0	2	2
2	0	1	1	3
4	1	1	2	4
5	0	0	4	4
7	0	1	1	1
8	0	1	2	4
10	0	0	3	3
11	0	1	1	4
12	0	1	3	4
13	0	1	3	3
14	0	1	3	1
15	0	0	2	4
16	0	1	1	1

	Attrition	Gender	Education	EnvironmentSatisfaction
EmployeeNumber				

[89]
```
# First, perform random forest the way it is, and can perform with pd.get
from sklearn.ensemble import RandomForestClassifier
```

[90]
```
rf_data = pd.concat([number_data, categorical_data], axis=1)
rf_data.head()
```

	Age	DailyRate	DistanceFromHome	HourlyRate	Jo
EmployeeNumber					
1	41	1102	1	94	2
2	49	279	8	61	2
4	37	1373	2	92	1
5	33	1392	3	56	1
7	27	591	2	40	1

5 rows × 26 columns

[91]
```
X = rf_data.drop(['Attrition', 'kmeans_2'], axis=1)
y = rf_data['Attrition']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

[92]
```
feat_importance = sorted(list(zip(X_train.columns, model.feature_importan
feat_importance
```

```
[('MonthlyIncome', 0.09221759374270919),
 ('Age', 0.07572211906201884),
 ('MonthlyRate', 0.07079060397330715),
 ('DailyRate', 0.0616860397684439),
 ('TotalWorkingYears', 0.06063186916853511),
 ('HourlyRate', 0.056199462963534445),
 ('DistanceFromHome', 0.05610164079114587),
 ('YearsAtCompany', 0.04739026724019447),
 ('NumCompaniesWorked', 0.04687023961174577),
```

```
('StockOptionLevel', 0.046660510696115785),
('PercentSalaryHike', 0.04194180334275924),
('YearsWithCurrManager', 0.03713519254303363),
('YearsInCurrentRole', 0.035898739340588424),
('TrainingTimesLastYear', 0.034396657027028356),
('JobSatisfaction', 0.03038047410486444),
('EnvironmentSatisfaction', 0.030202696194253663),
('YearsSinceLastPromotion', 0.029974154097946046),
('JobInvolvement', 0.0279880649489735),
('RelationshipSatisfaction', 0.026133871359906267),
('JobLevel', 0.025900062637684105),
('Education', 0.02574314521326226),
('WorkLifeBalance', 0.02347957821063492),
('Gender', 0.012198995995540427),
('PerformanceRating', 0.0043562179657742565)]
```

The results obtained from the calculation (n the order of the importance) is as follows:
(1) 'MonthlyIncome', 0.09620026433543806,
(2) 'Age', 0.07465100132268015,
(3) 'MonthlyRate', 0.0678728429991805,
(4) 'DailyRate', 0.06395539641367105,
(5) 'HourlyRate', 0.060778294100276326,
(6) 'TotalWorkingYears', 0.059386776367443835,
(7) 'DistanceFromHome', 0.05537957826054828,
(8) 'YearsAtCompany', 0.04897909582304177,
(9) 'StockOptionLevel', 0.04782603253763362,
(10) 'PercentSalaryHike', 0.04393563135475805,
(11) 'NumCompaniesWorked', 0.04368879940471908,
(12) 'YearsWithCurrManager', 0.037408724867448895,
(13) 'YearsInCurrentRole', 0.036640647371725926,
(14) 'TrainingTimesLastYear', 0.034674160760093234,
(15) 'JobSatisfaction', 0.0322863585903226,
(16) 'EnvironmentSatisfaction', 0.03193999230375698,
(17) 'YearsSinceLastPromotion', 0.029594830954866907,
(18) 'RelationshipSatisfaction', 0.027932283322458836,
(19) 'JobInvolvement', 0.025080256516368957,
(20) 'Education', 0.025006182546777035,
(21) 'WorkLifeBalance', 0.02088513349853633,
(22) 'JobLevel', 0.020547005053445814,
(23) 'Gender', 0.011329291160433025,
(24) 'PerformanceRating', 0.004021420134374657

Finally, the accuracy score is calculated to see if the results are reliable.

[93]
```
model.score(X_train, y_train), model.score(X_test, y_test)
```

```
(1.0, 0.8503401360544217)
```

Result: The accuracy score indicates that 0.85 (85%) is classified correctly, which is a high score knowing the score higher than 50% is considered a good score.

Although Random Forests above might have done a good job at detecting interactions between different features, if there are highly correlated features in the dataset, it can mask potential feature interactions.

As such, based on the correlation heatmap plotted earlier, the numeric_data dataframe's attributes are highly correlated with one another in many cases. Thus, the highly correlated columns (correation >= 0.5) will be removed to rerun the Random Forest Classifer.

Definitions used to define high correlation: Pearson's Correlation Coefficient
- Perfect: If the value is near ± 1, then it said to be a perfect correlation: as one variable increases, the other variable tends to also increase (if positive) or decrease (if negative).
- High degree: If the coefficient value lies between ± 0.50 and ± 1, then it is said to be a strong correlation.
- Moderate degree: If the value lies between ± 0.30 and ± 0.49, then it is said to be a medium correlation.
- Low degree: When the value lies below + .29, then it is said to be a small correlation.
- No correlation: When the value is zero.

---

[94]
```
# Create correlation matrix
corr_matrix = rf_data.corr().abs()
corr_matrix
```

	Age	DailyRate	DistanceFromHome	Hourly
Age	1.000000	0.010661	0.001686	0.0242
DailyRate	0.010661	1.000000	0.004985	0.0233
DistanceFromHome	0.001686	0.004985	1.000000	0.0311
HourlyRate	0.024287	0.023381	0.031131	1.0000
JobLevel	0.509604	0.002966	0.005303	0.0278
MonthlyIncome	0.497855	0.007707	0.017014	0.0157
MonthlyRate	0.028051	0.032182	0.027473	0.0152
NumCompaniesWorked	0.299635	0.038153	0.029251	0.0221
PercentSalaryHike	0.003634	0.022704	0.040235	0.0090
StockOptionLevel	0.037510	0.042143	0.044872	0.0502
TotalWorkingYears	0.680381	0.014515	0.004628	0.0023
TrainingTimesLastYear	0.019621	0.002453	0.036942	0.0085
YearsAtCompany	0.311309	0.034055	0.009508	0.0195
YearsInCurrentRole	0.212901	0.009932	0.018845	0.0241
YearsSinceLastPromotion	0.216513	0.033229	0.010029	0.0267

	Age	DailyRate	DistanceFromHome	Hourly

[95]
```
# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype
upper
```

	Age	DailyRate	DistanceFromHome	HourlyRat
Age	NaN	0.010661	0.001686	0.024287
DailyRate	NaN	NaN	0.004985	0.023381
DistanceFromHome	NaN	NaN	NaN	0.031131
HourlyRate	NaN	NaN	NaN	NaN
JobLevel	NaN	NaN	NaN	NaN
MonthlyIncome	NaN	NaN	NaN	NaN
MonthlyRate	NaN	NaN	NaN	NaN
NumCompaniesWorked	NaN	NaN	NaN	NaN
PercentSalaryHike	NaN	NaN	NaN	NaN
StockOptionLevel	NaN	NaN	NaN	NaN
TotalWorkingYears	NaN	NaN	NaN	NaN
TrainingTimesLastYear	NaN	NaN	NaN	NaN
YearsAtCompany	NaN	NaN	NaN	NaN
YearsInCurrentRole	NaN	NaN	NaN	NaN
YearsSinceLastPromotion	NaN	NaN	NaN	NaN
YearsWithCurrManager	NaN	NaN	NaN	NaN

[96]
```
# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.50
to_drop
```

```
['JobLevel',
 'MonthlyIncome',
 'TotalWorkingYears',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager',
 'PerformanceRating']
```

[97]
```
rf_data_feat_dropped = rf_data.drop(columns=to_drop)
```

[98]
```
#X = rf_data_feat_dropped.drop(['Attrition', 'kmeans_2'], axis=1)
#y = rf_data_feat_dropped['Attrition']
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train1, y_train1)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

[99]
```
new_feat_importance = sorted(list(zip(X_train1.columns, model.feature_imp
new_feat_importance
```

```
[('MonthlyIncome', 0.09224663310242301),
 ('Age', 0.0779918410189193),
 ('DailyRate', 0.06567910944877534),
 ('MonthlyRate', 0.06070814610847105),
 ('TotalWorkingYears', 0.05950168235375984),
 ('HourlyRate', 0.05707157360038054),
 ('DistanceFromHome', 0.05619221305673398),
 ('StockOptionLevel', 0.04991711761410612),
 ('YearsAtCompany', 0.04923976370903436),
 ('NumCompaniesWorked', 0.04666683318405298),
 ('PercentSalaryHike', 0.04543804063013404),
 ('YearsWithCurrManager', 0.03914546807892215),
 ('TrainingTimesLastYear', 0.03561476869129553),
 ('YearsInCurrentRole', 0.03414899956529789),
 ('JobSatisfaction', 0.03164052535313636),
 ('EnvironmentSatisfaction', 0.030145276398590503),
 ('YearsSinceLastPromotion', 0.02893178270367277),
 ('RelationshipSatisfaction', 0.026323924811830962),
 ('JobInvolvement', 0.025850228250203142),
 ('WorkLifeBalance', 0.024549239496501098),
 ('Education', 0.022423324479061785),
 ('JobLevel', 0.02090667731547117),
 ('Gender', 0.01413498150007824),
 ('PerformanceRating', 0.005531849529147735)]
```

[100]
```
model.score(X_train1, y_train1), model.score(X_test, y_test)
```

```
(1.0, 0.8480725623582767)
```

Results: After Dropping some highly correlated columns, the accuracy score increased from 0.8571 to 0.8613.

The feature importance above includes income-related columns though MonthlhIncome had been removed already. To check if such income-related columns may affect the accuracy score,

DailyRate, MonthlyRate, and HourlyRate will be removed to re-execute Random Forest classification again.

[101]
```
rf_data_feat_dropped2 = rf_data_feat_dropped.drop(columns=['DailyRate', '
rf_data_feat_dropped2
```

	Age	DistanceFromHome	NumCompaniesWorked	Perc
EmployeeNumber				
1	41	1	8	11
2	49	8	1	23
4	37	2	6	15
5	33	3	1	11
7	27	2	9	12
8	32	2	0	13
10	59	3	4	20
11	30	24	1	22
12	38	23	0	21
13	36	27	6	13
14	35	16	0	13
15	29	15	0	12
16	31	26	1	17
18	34	19	0	11
19	28	24	5	14

Empty markdown cell, double click me to add content.

[102]
```
#X = rf_data_feat_dropped2.drop(['Attrition', 'kmeans_2'], axis=1)
#y = rf_data_feat_dropped2['Attrition']
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train2, y_train2)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
```

```
                oob_score=False, random_state=None, verbose=0,
                warm_start=False)
```

[103]
```
new_feat_importance2 = sorted(list(zip(X_train2.columns, model.feature_im
new_feat_importance2
```

```
[('MonthlyIncome', 0.08577197384941267),
 ('Age', 0.07939721618966501),
 ('DailyRate', 0.06526541654094985),
 ('MonthlyRate', 0.06286140924252112),
 ('TotalWorkingYears', 0.05987624844014459),
 ('HourlyRate', 0.05890154009971002),
 ('DistanceFromHome', 0.05436664984236051),
 ('StockOptionLevel', 0.049949689449122696),
 ('YearsAtCompany', 0.04868478456253448),
 ('NumCompaniesWorked', 0.0482021945185903),
 ('PercentSalaryHike', 0.04275554023085288),
 ('YearsWithCurrManager', 0.04250664425585016),
 ('YearsInCurrentRole', 0.03437175786731954),
 ('EnvironmentSatisfaction', 0.03352283665315875),
 ('JobSatisfaction', 0.03242898921034155),
 ('TrainingTimesLastYear', 0.032335126846135084),
 ('YearsSinceLastPromotion', 0.028076988298994642),
 ('RelationshipSatisfaction', 0.026510215005519804),
 ('WorkLifeBalance', 0.026360911678767685),
 ('Education', 0.025212404389349293),
 ('JobInvolvement', 0.025184502710373848),
 ('JobLevel', 0.02105818126506836),
 ('Gender', 0.011743525333408071),
 ('PerformanceRating', 0.004655253519849054)]
```

[104]
```
model.score(X_train2, y_train2), model.score(X_test, y_test)
```

```
(1.0, 0.8526077097505669)
```

Results: After further dropping income related columns, the accuracy score increased from 0.8613 to 0.8639.