Week 10.2 Assignment

- Joshua Burden
- Bellevue University
- DSC550 Data Mining
- Dr. Brett Werner
- 11/06/2022

Begin Milestone 1 with a 250-500-word narrative describing your original idea for the analysis/model building business problem.

Clearly identify the problem you will address and the target for your model.

Background:

During the pandemic an increase in the need for health care professionals was required. The dataset collected is a modified synthetic dataset from IBM's Watson to show a useful insight into the attrition rate for healthcare workers.

Problem:

The data set includes information about the attrition rate for employees within the healthcare field. The meaning of employee attrition is the departure of employees from the organization for any reason whether that be voluntary or involuntary, including resignation, termination, death, or retirement. Companies to avoid attrition rates being too high is to replace those who are either leaving voluntarily or involuntary. The data set should provide insights into whether a company in the healthcare field was replacing their employees that were leaving the field, or if they continued to have a gradual but deliberate reduction in staff for any reason.

Original Idea:

The idea behind this data set is to discover whether certain roles within the healthcare industry, hours worked, age of an employee, or any other qualifying data points stand out as to why the healthcare industry had any determining factor on whether a person was to leave their field, while also predicting whether the employee was eventually replaced.

Dataset:

This dataset contains employee and company data useful for supervised ML, unsupervised ML, and analytics. Attrition - whether an employee left or not - is included and can be used as the target variable. The data is synthetic and based on the IBM Watson dataset for attrition. Employee roles and departments were changed to reflect the healthcare domain. Also, known outcomes for some employees were changed to help increase the performance of ML models

Then, do a graphical analysis creating a minimum of four graphs.

Label your graphs appropriately and explain/analyze the information provided by each graph.

In [ ]:
```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import numpy as np
import plotly.express as px
```

In [ ]:
```
data_df = pd.read_csv('./DATA/watson_healthcare_modified.csv')
data_df.head()
```

Out[ ]:

	EmployeeID	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educati
0	1313919	41	No	Travel_Rarely	1102	Cardiology	1	
1	1200302	49	No	Travel_Frequently	279	Maternity	8	
2	1060315	37	Yes	Travel_Rarely	1373	Maternity	2	
3	1272912	33	No	Travel_Frequently	1392	Maternity	3	
4	1414939	27	No	Travel_Rarely	591	Maternity	2	

5 rows × 35 columns

In [ ]:
```
print("Number of duplicated data: "+str(data_df.duplicated().sum()))
```

```
Number of duplicated data: 0
```

In [ ]:
```
data_df.isnull().sum()
```

Out[ ]:
```
EmployeeID                  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
Shift                       0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [ ]:
```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1676 entries, 0 to 1675
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   EmployeeID                1676 non-null   int64 
 1   Age                       1676 non-null   int64 
 2   Attrition                 1676 non-null   object
 3   BusinessTravel            1676 non-null   object
 4   DailyRate                 1676 non-null   int64 
 5   Department                1676 non-null   object
 6   DistanceFromHome          1676 non-null   int64 
 7   Education                 1676 non-null   int64 
 8   EducationField            1676 non-null   object
 9   EmployeeCount             1676 non-null   int64 
 10  EnvironmentSatisfaction   1676 non-null   int64 
 11  Gender                    1676 non-null   object
 12  HourlyRate                1676 non-null   int64 
 13  JobInvolvement            1676 non-null   int64 
 14  JobLevel                  1676 non-null   int64 
 15  JobRole                   1676 non-null   object
 16  JobSatisfaction           1676 non-null   int64 
 17  MaritalStatus             1676 non-null   object
 18  MonthlyIncome             1676 non-null   int64 
 19  MonthlyRate               1676 non-null   int64 
 20  NumCompaniesWorked        1676 non-null   int64 
 21  Over18                    1676 non-null   object
 22  OverTime                  1676 non-null   object
 23  PercentSalaryHike         1676 non-null   int64 
 24  PerformanceRating         1676 non-null   int64 
 25  RelationshipSatisfaction  1676 non-null   int64 
 26  StandardHours             1676 non-null   int64 
 27  Shift                     1676 non-null   int64 
 28  TotalWorkingYears         1676 non-null   int64 
 29  TrainingTimesLastYear     1676 non-null   int64 
 30  WorkLifeBalance           1676 non-null   int64 
 31  YearsAtCompany            1676 non-null   int64 
 32  YearsInCurrentRole        1676 non-null   int64 
 33  YearsSinceLastPromotion   1676 non-null   int64 
 34  YearsWithCurrManager      1676 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 458.4+ KB
```

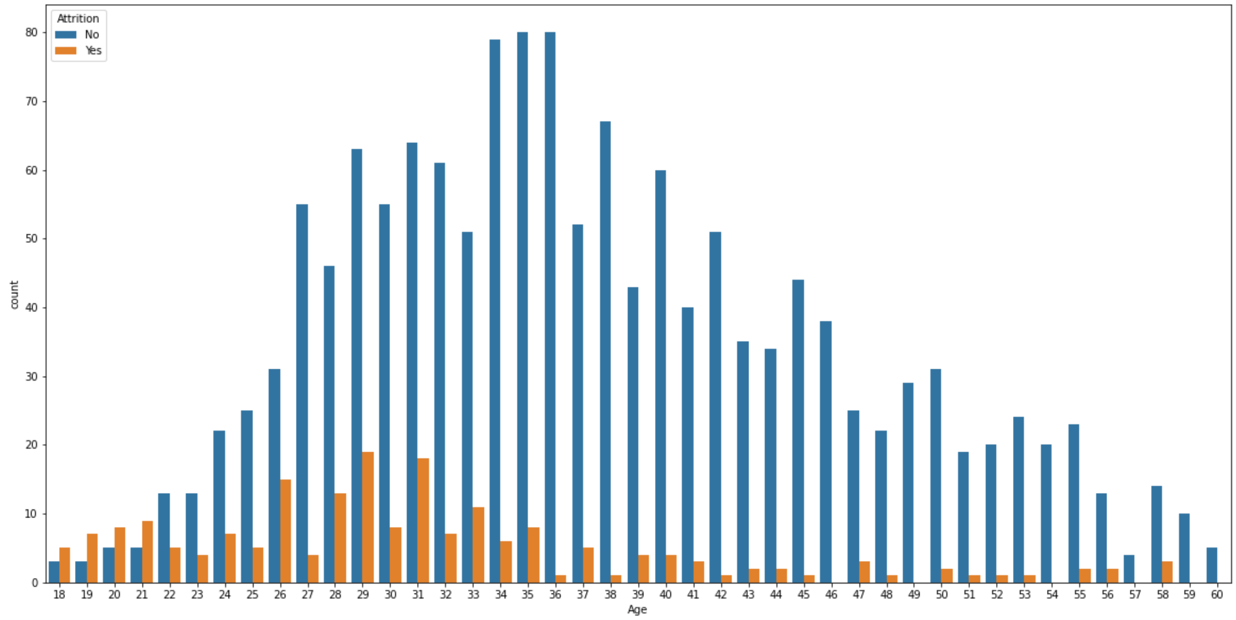
Visualization 1

In [ ]:

```
plt.figure(figsize=(20,10))
sns.countplot(x='Age',hue='Attrition',data=data_df)
```

Out[ ]:

```
<AxesSubplot:xlabel='Age', ylabel='count'>
```



Visualization 2

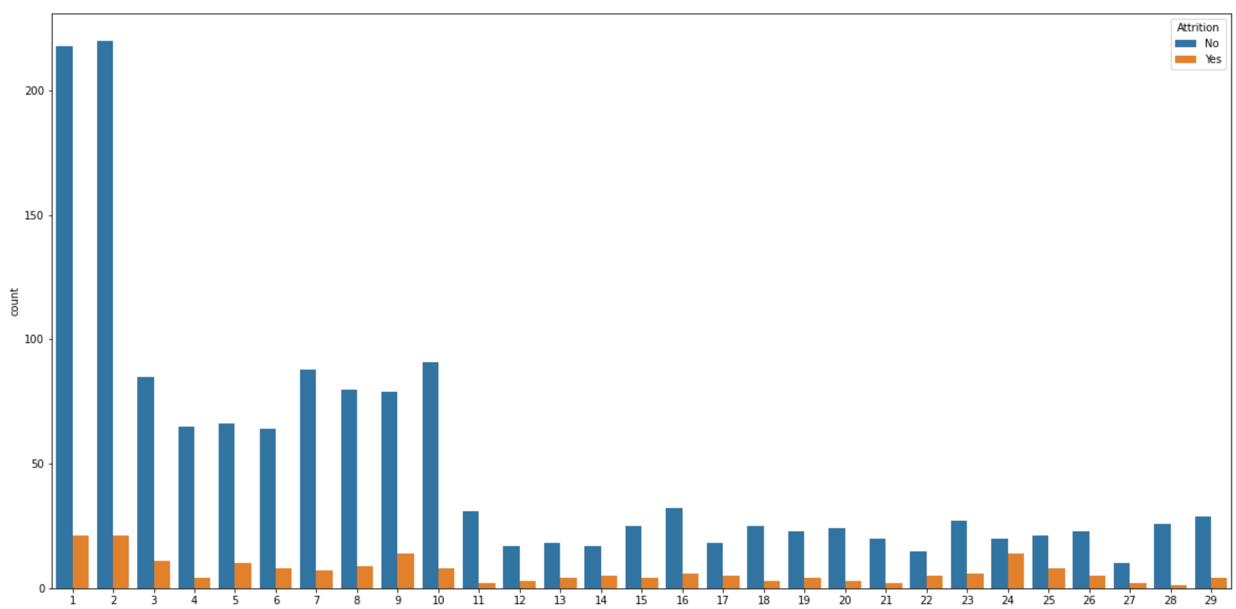
In [ ]:
```
data_df.groupby('Attrition')['MonthlyIncome'].mean().sort_values().reset_index()
```

Out[ ]:

	Attrition	MonthlyIncome
0	Yes	4024.246231
1	No	6852.301963

In [ ]:
```
plt.figure(figsize=(20,10))
sns.countplot(x='DistanceFromHome',hue='Attrition',data=data_df)
```

Out[ ]:
```
<AxesSubplot:xlabel='DistanceFromHome', ylabel='count'>
```



Visualization 3

In [ ]:
```
px.histogram(data_df,x="Department",color="Attrition",barmode="group",text_auto=".2f",
             title = "Percentage of Department Type")
```

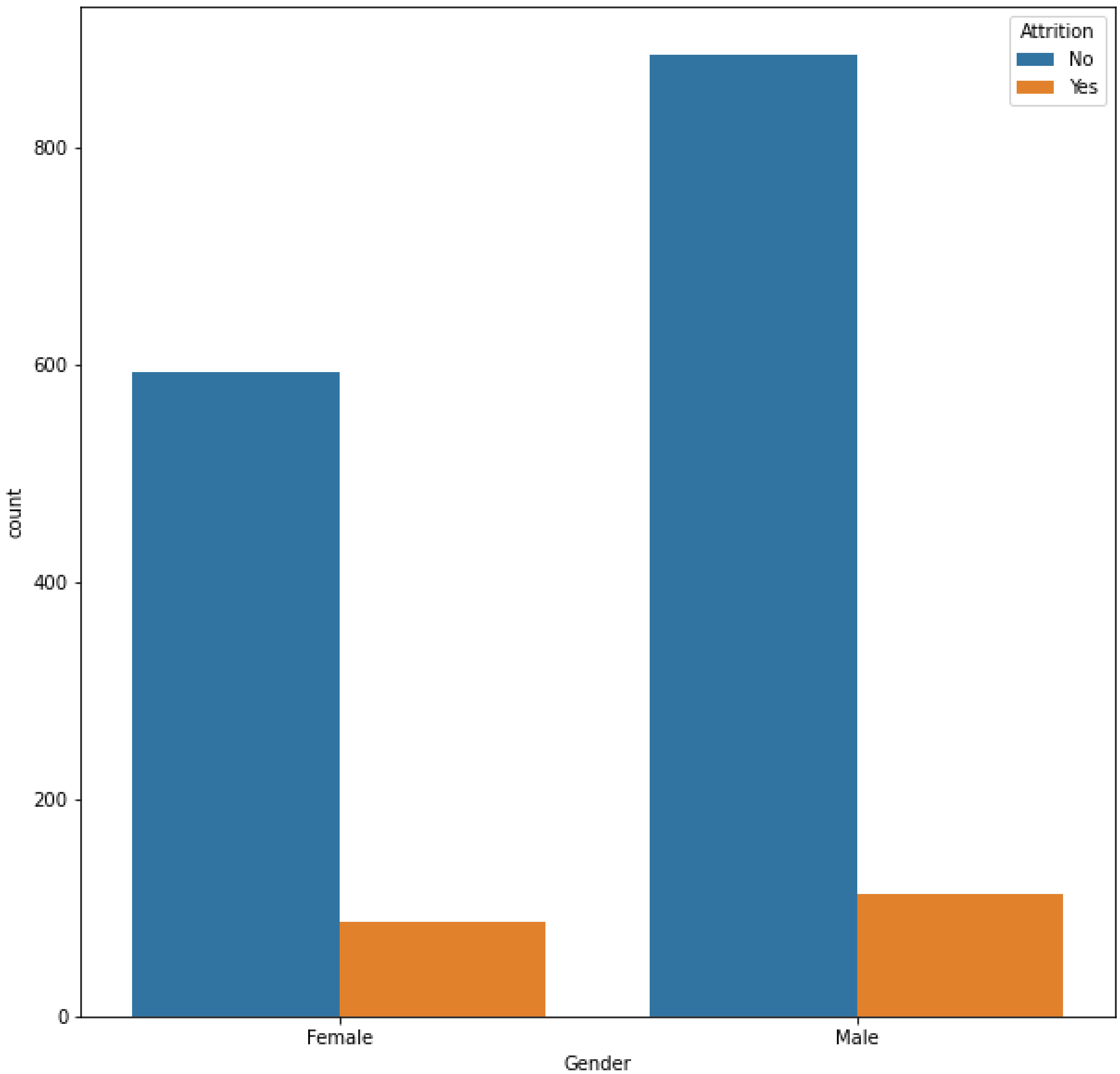
In [ ]:
```
px.histogram(data_df,x="EducationField",color="Attrition",barmode="group",text_auto=".
             title = "Percentage of EducationField Type")
```

In [ ]:
```
px.histogram(data_df,x="JobRole",color="Attrition",barmode="group",text_auto=".2f",tem
             title = "Percentage of EducationField Type")
```

Visualization 4

In [ ]:
```
plt.figure(figsize=(10,10))
sns.countplot(x='Gender',hue='Attrition',data=data_df)
```

Out[ ]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



Breakdown of all the available datapoints

In [ ]:
```
plt.figure(figsize=(30,50))
for index,column in enumerate(num_col):
    plt.subplot(5,5,index+1)
    sns.countplot(data=num_col,x=column)
    plt.xticks(rotation = 90)
plt.tight_layout(pad = 1.0)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
c:\Users\Joshu\Desktop\Masters\DSC550\JoshuaBurdenAssignment10-2.ipynb Cell 25 in <cell line: 2>()
      <a href='vscode-notebook-cell:/c%3A/Users/Joshu/Desktop/Masters/DSC550/JoshuaBurdenAssignment10-2.ipynb#X33sZmlsZQ%3D%3D?line=0'>1</a> plt.figure(figsize=(30,50))
----> <a href='vscode-notebook-cell:/c%3A/Users/Joshu/Desktop/Masters/DSC550/JoshuaBurdenAssignment10-2.ipynb#X33sZmlsZQ%3D%3D?line=1'>2</a> for index,column in enumerate(num_col):
      <a href='vscode-notebook-cell:/c%3A/Users/Joshu/Desktop/Masters/DSC550/JoshuaBurdenAssignment10-2.ipynb#X33sZmlsZQ%3D%3D?line=2'>3</a>     plt.subplot(5,5,index+1)
      <a href='vscode-notebook-cell:/c%3A/Users/Joshu/Desktop/Masters/DSC550/JoshuaBurdenAssignment10-2.ipynb#X33sZmlsZQ%3D%3D?line=3'>4</a>     sns.countplot(data=num_col,x=column)

NameError: name 'num_col' is not defined
<Figure size 2160x3600 with 0 Axes>
```

Observations:

- Maternity departments had the highest rate of attrition followed by cardiology and neurology
- attrition rates had the highest peak at 29 years old
- 26-35 years old saw the highest range of attrition
- 42 years old and older saw the least attrition rates
- More men where likely to leave than women but Men also were more accounted for than women in the healthcare field
- Human resources were the least likely to have people quit
- Life Sciences were the Education field with the highest amount of attrition
- people that lived closer to their jobs were more likely to leave

Milestone 2

In [ ]:
```
data_df.head()
```

Out[ ]:

	EmployeeID	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educati
0	1313919	41	No	Travel_Rarely	1102	Cardiology	1	
1	1200302	49	No	Travel_Frequently	279	Maternity	8	
2	1060315	37	Yes	Travel_Rarely	1373	Maternity	2	
3	1272912	33	No	Travel_Frequently	1392	Maternity	3	
4	1414939	27	No	Travel_Rarely	591	Maternity	2	

5 rows × 35 columns

In [ ]:
```
data_df.shape
```

Out[ ]:
```
(1676, 35)
```

Dropping data columns that don't provide much value or context to the data

In [ ]:
```
#drop some of the values
data_df1 = data_df.drop(['EmployeeID','Over18', 'EmployeeCount', 'StandardHours'], axi
data_df1.head()
```

Out[ ]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Education
0	41	No	Travel_Rarely	1102	Cardiology	1	2	Life Sci
1	49	No	Travel_Frequently	279	Maternity	8	1	Life Sci
2	37	Yes	Travel_Rarely	1373	Maternity	2	2	(
3	33	No	Travel_Frequently	1392	Maternity	3	4	Life Sci
4	27	No	Travel_Rarely	591	Maternity	2	1	M

5 rows × 31 columns

In [ ]:
```
#change attrition rates from no/yes to 0/1
data_df1['Attrition'] = data_df1['Attrition'].str.replace('Yes', str(1))
data_df1['Attrition'] = data_df1['Attrition'].str.replace('No', str(0))
data_df1['Attrition'] = data_df1['Attrition'].astype('int')
```

converted attriations yes/no values and replaced them with 0/1 and set type to int

Look at shape and values of columns

In [ ]:
```
data_df1.head()
```

Out[ ]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Education
0	41	0	Travel_Rarely	1102	Cardiology	1	2	Life Sci
1	49	0	Travel_Frequently	279	Maternity	8	1	Life Sci
2	37	1	Travel_Rarely	1373	Maternity	2	2	(
3	33	0	Travel_Frequently	1392	Maternity	3	4	Life Sci
4	27	0	Travel_Rarely	591	Maternity	2	1	M

5 rows × 31 columns

In [ ]:

```
data_df1.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	ma
Age	1676.0	36.866348	9.129126	18.0	30.00	36.0	43.00	60
Attrition	1676.0	0.118735	0.323573	0.0	0.00	0.0	0.00	1
DailyRate	1676.0	800.557876	401.594438	102.0	465.00	796.5	1157.00	1499
DistanceFromHome	1676.0	9.221957	8.158118	1.0	2.00	7.0	14.00	29
Education	1676.0	2.907518	1.025835	1.0	2.00	3.0	4.00	5
EnvironmentSatisfaction	1676.0	2.714797	1.097534	1.0	2.00	3.0	4.00	4
HourlyRate	1676.0	65.470167	20.207572	30.0	48.00	65.5	83.00	100
JobInvolvement	1676.0	2.724940	0.714121	1.0	2.00	3.0	3.00	4
JobLevel	1676.0	2.066826	1.113423	1.0	1.00	2.0	3.00	5
JobSatisfaction	1676.0	2.738663	1.104005	1.0	2.00	3.0	4.00	4
MonthlyIncome	1676.0	6516.512530	4728.456618	1009.0	2928.25	4899.0	8380.25	19999
MonthlyRate	1676.0	14287.019690	7138.857783	2094.0	7993.00	14269.5	20462.00	26999
NumCompaniesWorked	1676.0	2.662291	2.477704	0.0	1.00	2.0	4.00	9
PercentSalaryHike	1676.0	15.196897	3.646550	11.0	12.00	14.0	18.00	25
PerformanceRating	1676.0	3.150358	0.357529	3.0	3.00	3.0	3.00	4
RelationshipSatisfaction	1676.0	2.718377	1.078162	1.0	2.00	3.0	4.00	4
Shift	1676.0	0.806086	0.855527	0.0	0.00	1.0	1.00	3
TotalWorkingYears	1676.0	11.338902	7.834996	0.0	6.00	10.0	15.00	40
TrainingTimesLastYear	1676.0	2.805489	1.288431	0.0	2.00	3.0	3.00	6
WorkLifeBalance	1676.0	2.766110	0.702369	1.0	2.00	3.0	3.00	4
YearsAtCompany	1676.0	7.033413	6.098991	0.0	3.00	5.0	10.00	40
YearsInCurrentRole	1676.0	4.264916	3.627456	0.0	2.00	3.0	7.00	18
YearsSinceLastPromotion	1676.0	2.200477	3.229587	0.0	0.00	1.0	3.00	15
YearsWithCurrManager	1676.0	4.135442	3.559662	0.0	2.00	3.0	7.00	17

check correlation of data columns

In [ ]:

```
data_df1.corr()
```

Out[ ]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	Environment
Age	1.000000	-0.239984	0.001441	-0.010079	0.204655	
Attrition	-0.239984	1.000000	-0.053892	0.105580	-0.038843	
DailyRate	0.001441	-0.053892	1.000000	-0.009227	-0.015881	
DistanceFromHome	-0.010079	0.105580	-0.009227	1.000000	0.015937	
Education	0.204655	-0.038843	-0.015881	0.015937	1.000000	
EnvironmentSatisfaction	0.008945	-0.101278	0.010620	-0.019730	-0.031925	
HourlyRate	0.034671	-0.036300	0.027128	0.026947	0.017996	
JobInvolvement	0.034193	-0.166036	0.058864	0.010281	0.041046	
JobLevel	0.518333	-0.207634	0.009005	-0.023455	0.093227	
JobSatisfaction	-0.015848	-0.081881	0.032115	-0.004758	-0.003957	
MonthlyIncome	0.511378	-0.193527	0.011030	-0.041201	0.085116	
MonthlyRate	0.025837	0.045744	-0.032211	0.031672	-0.019198	
NumCompaniesWorked	0.296045	0.017279	0.034296	-0.024969	0.126758	
PercentSalaryHike	0.007570	0.002943	0.019325	0.034172	-0.006461	
PerformanceRating	0.005246	0.010728	0.003353	0.020482	-0.020664	
RelationshipSatisfaction	0.058528	-0.020462	0.014539	0.005482	-0.005750	
Shift	0.037117	-0.158322	0.054407	0.029180	0.024451	
TotalWorkingYears	0.692512	-0.234182	0.009378	-0.017663	0.143324	
TrainingTimesLastYear	-0.015408	-0.054836	0.001901	-0.055471	-0.014070	
WorkLifeBalance	-0.004878	-0.090513	-0.028549	-0.037821	0.003933	
YearsAtCompany	0.319012	-0.201373	-0.026892	-0.007420	0.057461	
YearsInCurrentRole	0.222655	-0.207891	0.019651	0.011448	0.051029	
YearsSinceLastPromotion	0.217212	-0.086207	-0.034571	-0.000126	0.045785	
YearsWithCurrManager	0.215909	-0.201087	-0.025272	0.000403	0.055096	

24 rows × 24 columns

dropping rows that exceed a threshold of 0.2

In [ ]:

```
threshold = 0.2
data_df1 = data_df1.drop(data_df1.std()[data_df1.std() < threshold].index.values, axis
```

```
C:\Users\Joshu\AppData\Local\Temp\ipykernel_316\2220865994.py:2: FutureWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
```

In [ ]:
```
data_df1
```

Out[ ]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
0	41	0	Travel_Rarely	1102	Cardiology	1	2	Life
1	49	0	Travel_Frequently	279	Maternity	8	1	Life
2	37	1	Travel_Rarely	1373	Maternity	2	2	
3	33	0	Travel_Frequently	1392	Maternity	3	4	Life
4	27	0	Travel_Rarely	591	Maternity	2	1	
...	...	...	...	...	...	...	...	
1671	26	1	Travel_Rarely	471	Neurology	24	3	
1672	46	0	Travel_Rarely	1125	Cardiology	10	3	M
1673	20	0	Travel_Rarely	959	Maternity	1	3	Life
1674	39	0	Travel_Rarely	466	Neurology	1	1	Life
1675	27	0	Travel_Rarely	511	Cardiology	2	2	

1676 rows × 31 columns

In [ ]:
```
#Get dummies
data_df2 = data_df1.copy()
data_df2 = pd.get_dummies(data_df2, drop_first=True)
data_df2
```

Out[ ]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate
0	41	0	1102	1	2	2	94
1	49	0	279	8	1	3	61
2	37	1	1373	2	2	4	92
3	33	0	1392	3	4	4	56
4	27	0	591	2	1	1	40
...	...	...	...	...	...	...	...
1671	26	1	471	24	3	3	66
1672	46	0	1125	10	3	3	94
1673	20	0	959	1	3	4	83
1674	39	0	466	1	1	4	65
1675	27	0	511	2	2	1	89

1676 rows × 41 columns

In [ ]:
```
data_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1676 entries, 0 to 1675
Data columns (total 41 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Age                                1676 non-null   int64
 1   Attrition                          1676 non-null   int32
 2   DailyRate                          1676 non-null   int64
 3   DistanceFromHome                   1676 non-null   int64
 4   Education                          1676 non-null   int64
 5   EnvironmentSatisfaction            1676 non-null   int64
 6   HourlyRate                         1676 non-null   int64
 7   JobInvolvement                     1676 non-null   int64
 8   JobLevel                           1676 non-null   int64
 9   JobSatisfaction                    1676 non-null   int64
 10  MonthlyIncome                      1676 non-null   int64
 11  MonthlyRate                        1676 non-null   int64
 12  NumCompaniesWorked                 1676 non-null   int64
 13  PercentSalaryHike                  1676 non-null   int64
 14  PerformanceRating                  1676 non-null   int64
 15  RelationshipSatisfaction           1676 non-null   int64
 16  Shift                              1676 non-null   int64
 17  TotalWorkingYears                  1676 non-null   int64
 18  TrainingTimesLastYear              1676 non-null   int64
 19  WorkLifeBalance                    1676 non-null   int64
 20  YearsAtCompany                     1676 non-null   int64
 21  YearsInCurrentRole                 1676 non-null   int64
 22  YearsSinceLastPromotion            1676 non-null   int64
 23  YearsWithCurrManager               1676 non-null   int64
 24  BusinessTravel_Travel_Frequently   1676 non-null   uint8
 25  BusinessTravel_Travel_Rarely       1676 non-null   uint8
 26  Department_Maternity               1676 non-null   uint8
 27  Department_Neurology               1676 non-null   uint8
 28  EducationField_Life Sciences       1676 non-null   uint8
 29  EducationField_Marketing           1676 non-null   uint8
 30  EducationField_Medical             1676 non-null   uint8
 31  EducationField_Other               1676 non-null   uint8
 32  EducationField_Technical Degree    1676 non-null   uint8
 33  Gender_Male                        1676 non-null   uint8
 34  JobRole_Administrative             1676 non-null   uint8
 35  JobRole_Nurse                      1676 non-null   uint8
 36  JobRole_Other                      1676 non-null   uint8
 37  JobRole_Therapist                  1676 non-null   uint8
 38  MaritalStatus_Married              1676 non-null   uint8
 39  MaritalStatus_Single               1676 non-null   uint8
 40  OverTime_Yes                       1676 non-null   uint8
dtypes: int32(1), int64(23), uint8(17)
memory usage: 335.7 KB
```

create the final dataframe

In [ ]:
```
df_final = data_df2.copy()
df_final
```

Out[ ]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate
0	41	0	1102	1	2	2	94
1	49	0	279	8	1	3	61
2	37	1	1373	2	2	4	92
3	33	0	1392	3	4	4	56
4	27	0	591	2	1	1	40
...	...	...	...	...	...	...	...
1671	26	1	471	24	3	3	66
1672	46	0	1125	10	3	3	94
1673	20	0	959	1	3	4	83
1674	39	0	466	1	1	4	65
1675	27	0	511	2	2	1	89

1676 rows × 41 columns

Create a training and test model set

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

In [ ]:
```
X = df_final.drop(['Attrition'], axis='columns')
```

In [ ]:
```
y= df_final['Attrition']
y.to_frame().head()
```

Out[ ]:

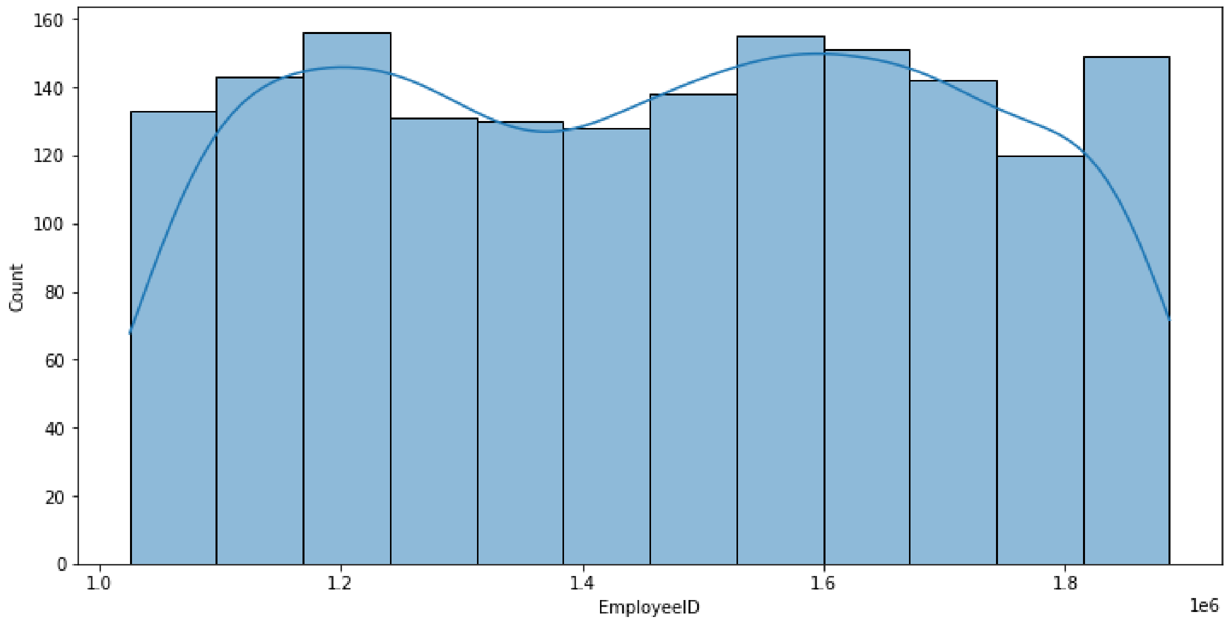
	Attrition
0	0
1	0
2	1
3	0
4	0

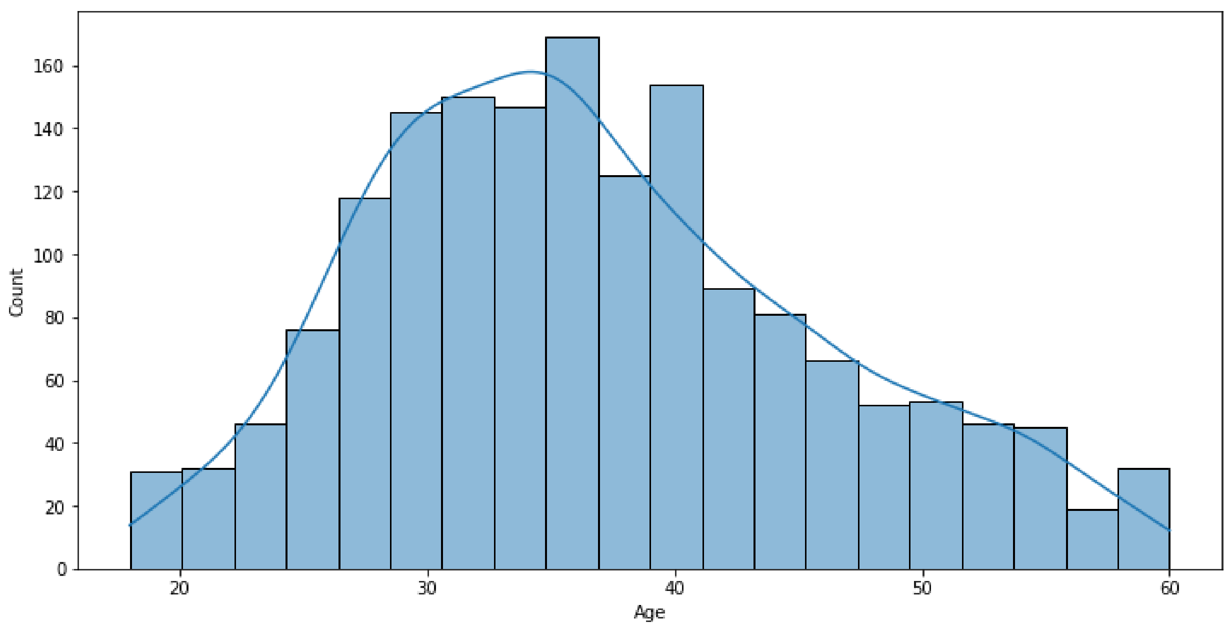
In [ ]:
```
print("X shape",X.shape,"\n","y shape",y.shape)
```

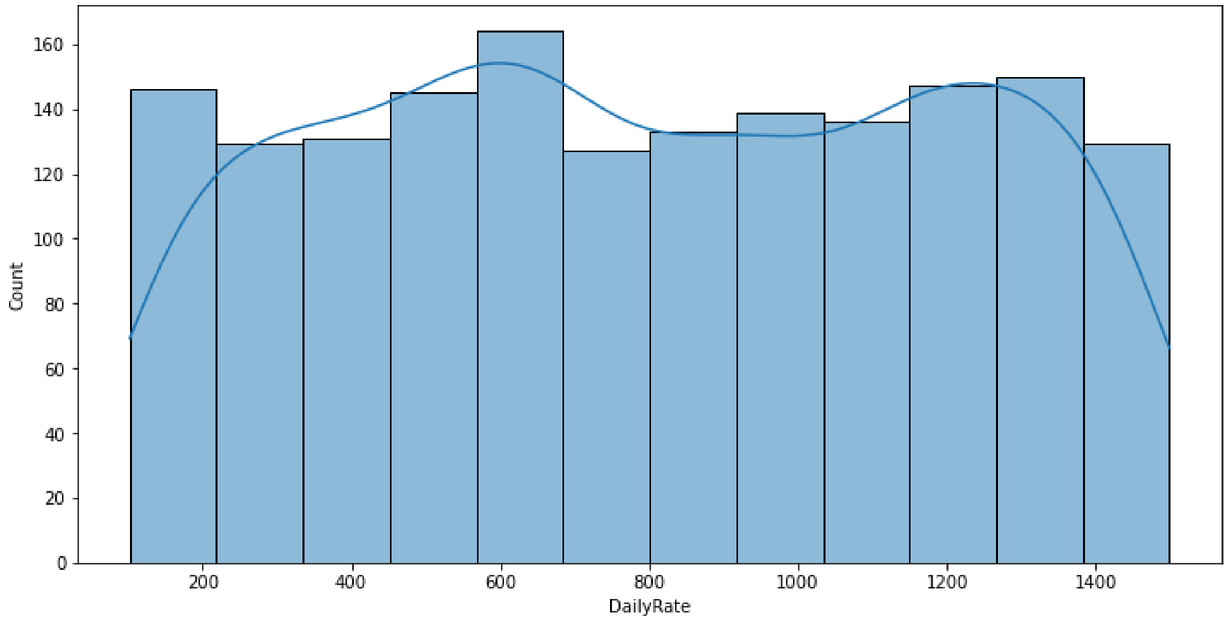
```
X shape (1676, 40)
 y shape (1676,)
```

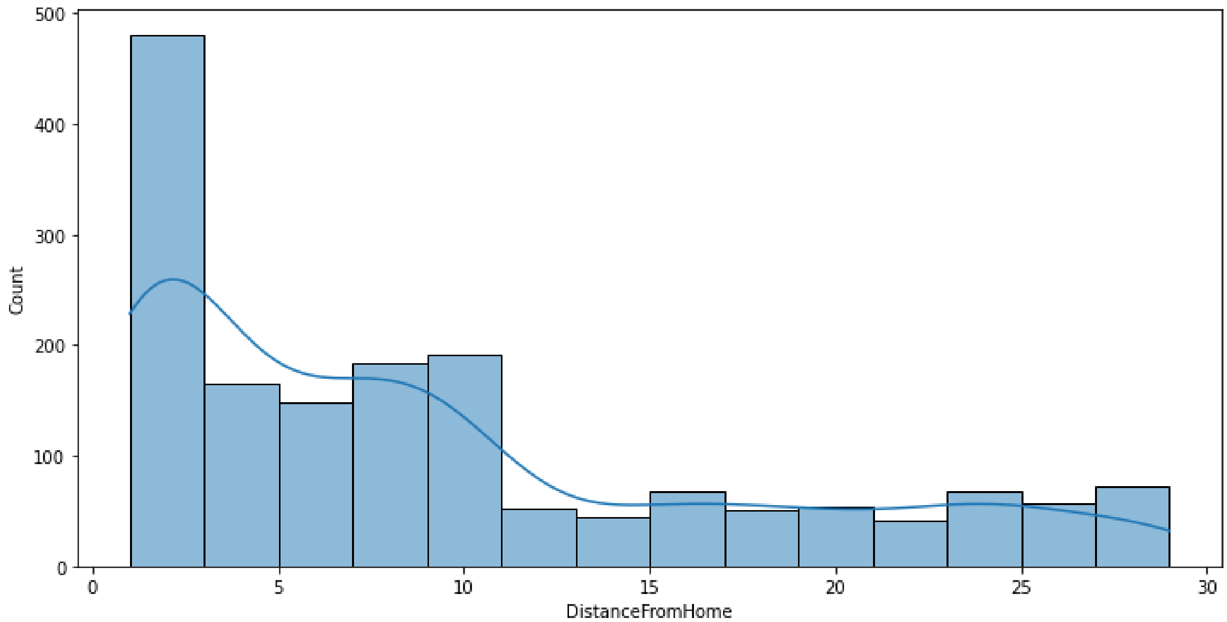
In [ ]:
```
num_cols = [col for col in data_df.columns if data_df[col].dtypes in ("int64", "float6
```

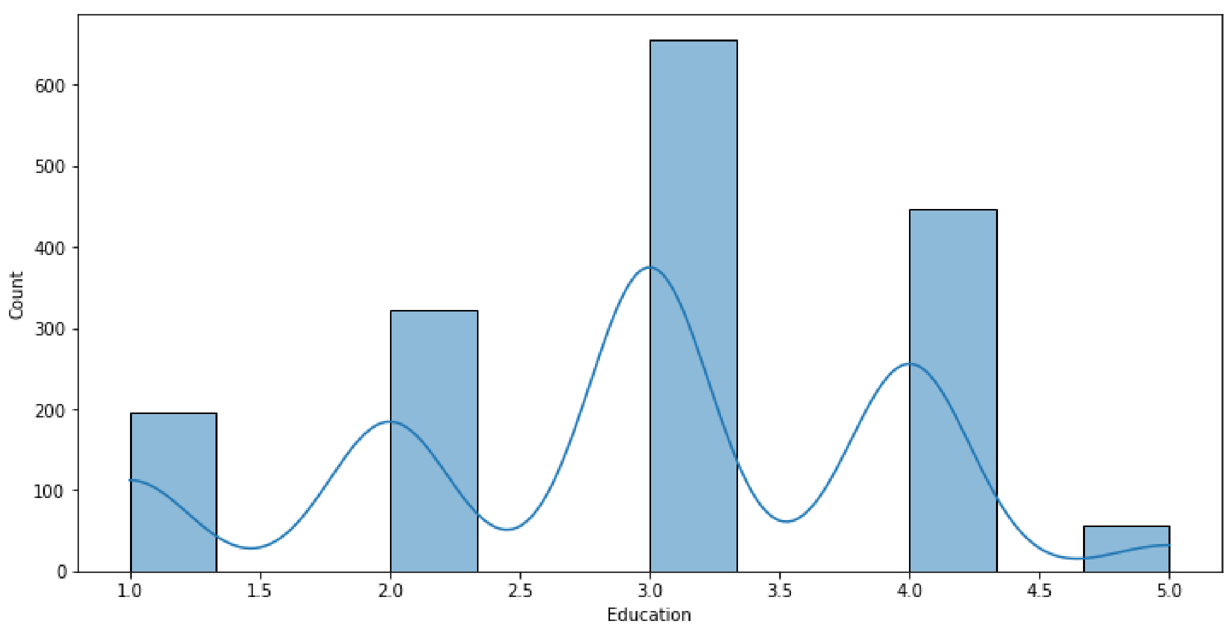
In [ ]:
```
for col in num_cols:
    fig, ax = plt.subplots(1, figsize = (12, 6))
    sns.histplot(data = data_df[col], kde = True)
    plt.show()
```

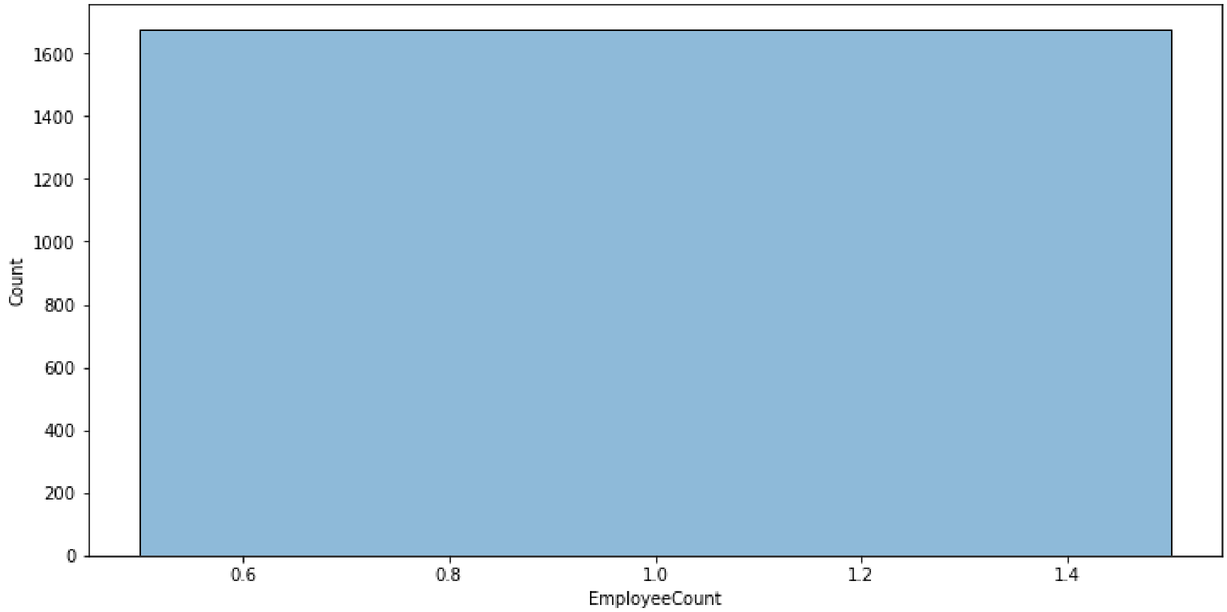


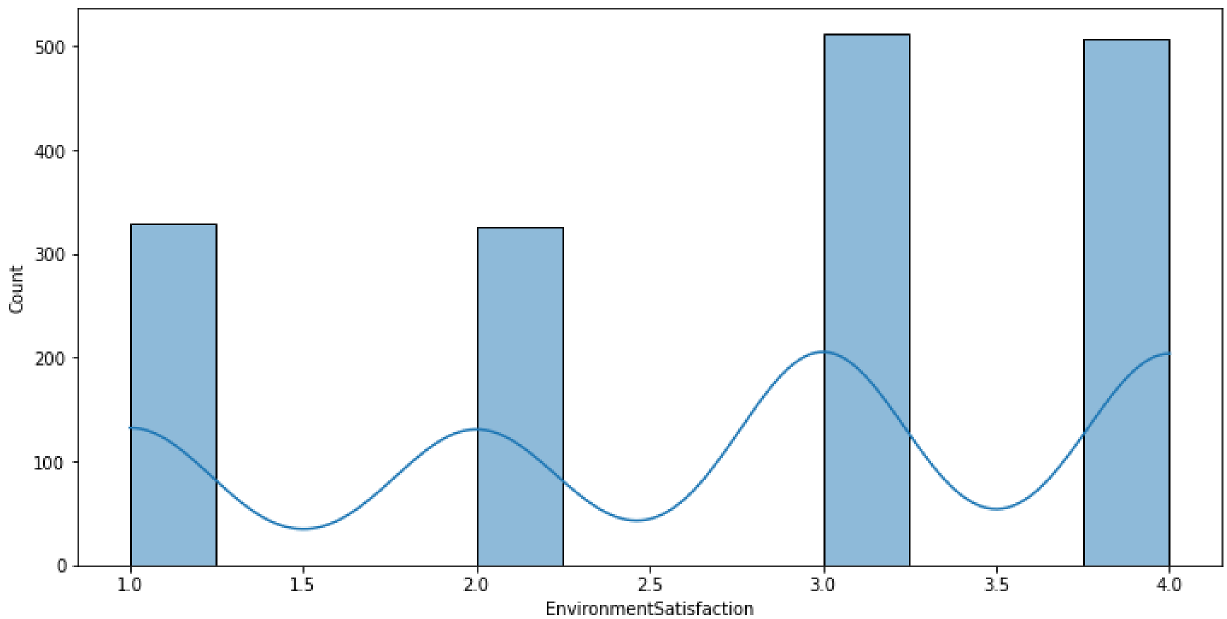


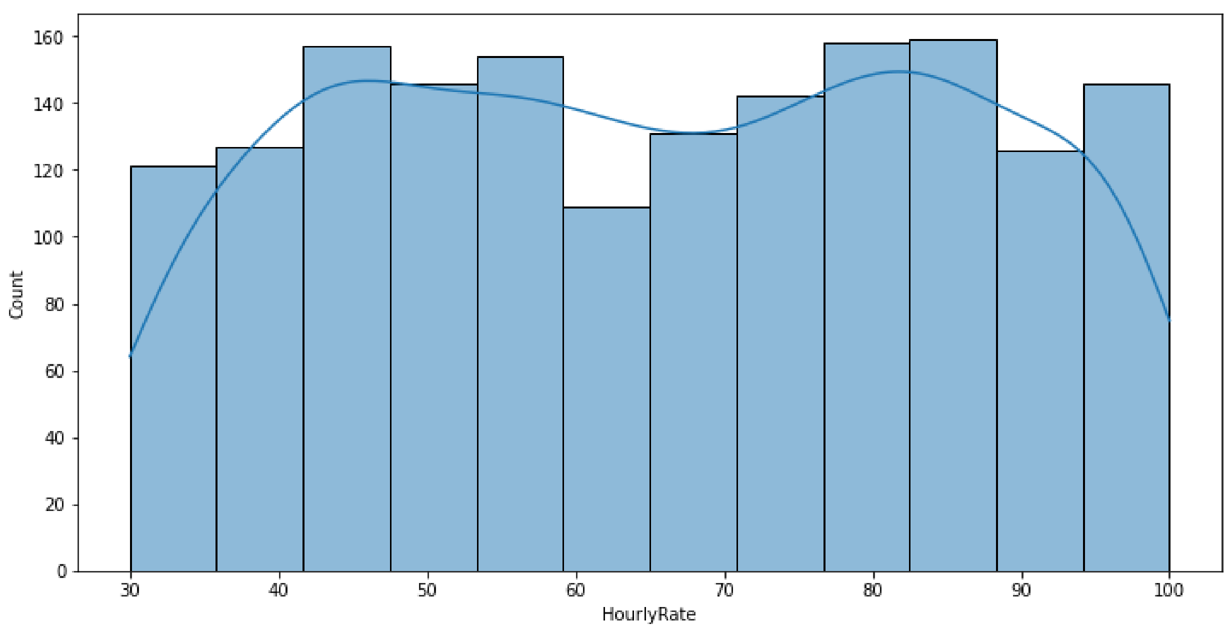

Count
0
20
40
60
80
100
120
140
160
200
400
600
800
1000
1200
1400
DailyRate

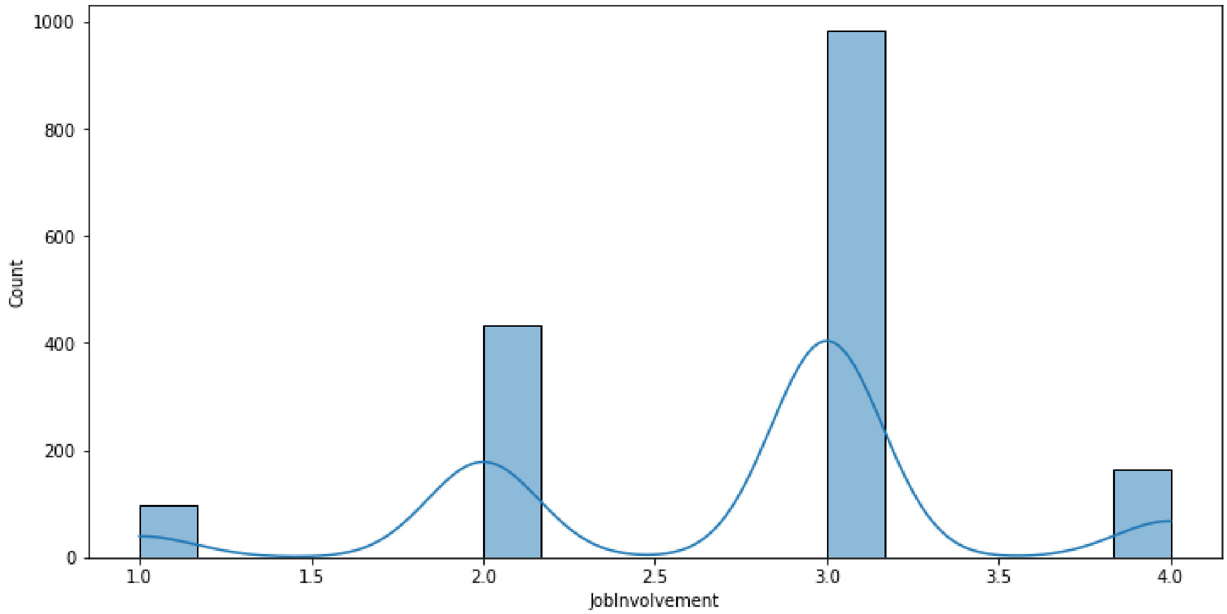

Count
0
100
200
300
400
500
0
5
10
15
20
25
30
DistanceFromHome

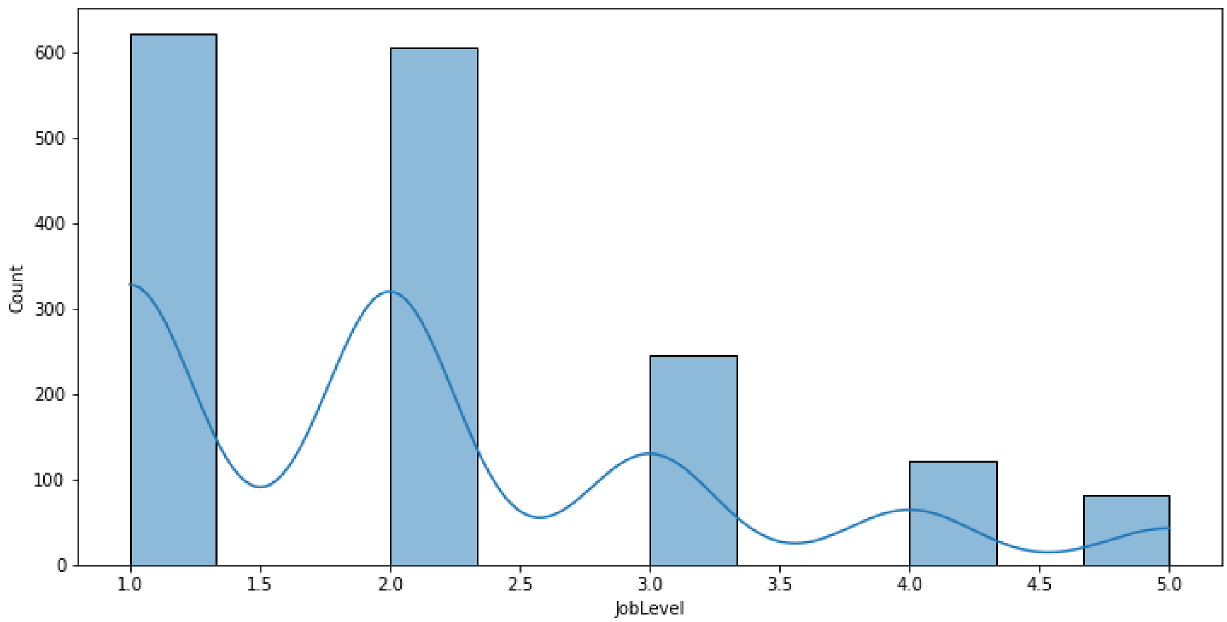

Count
0
100
200
300
400
500
600
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Education

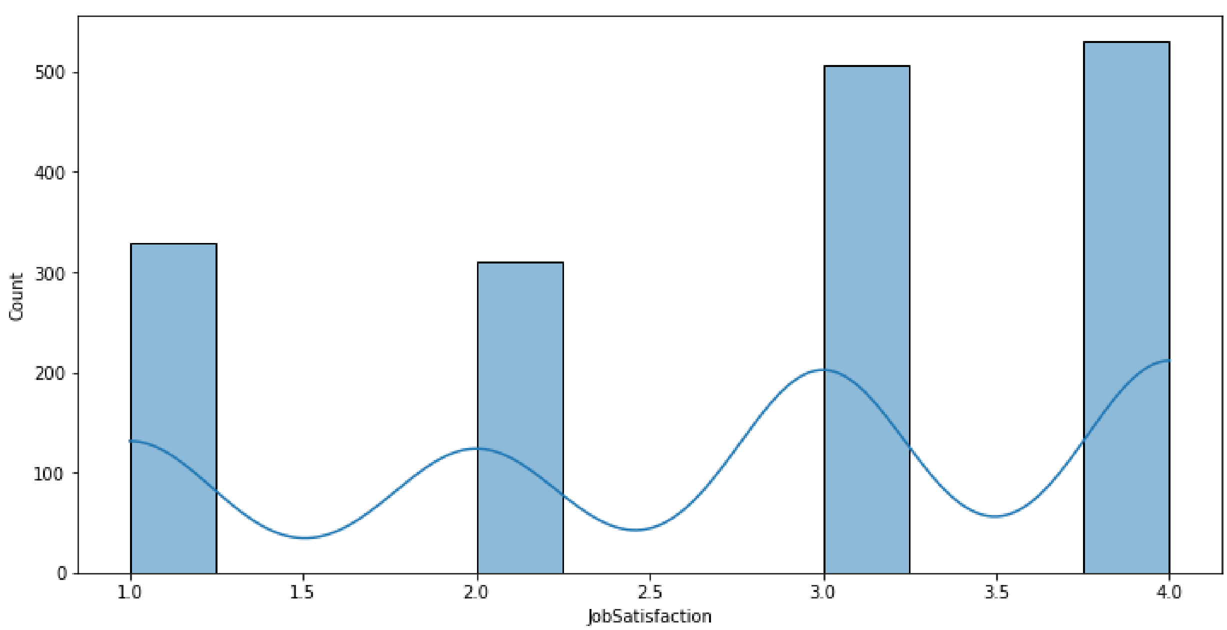

Count
EmployeeCount
0
200
400
600
800
1000
1200
1400
1600
0.6
0.8
1.0
1.2
1.4

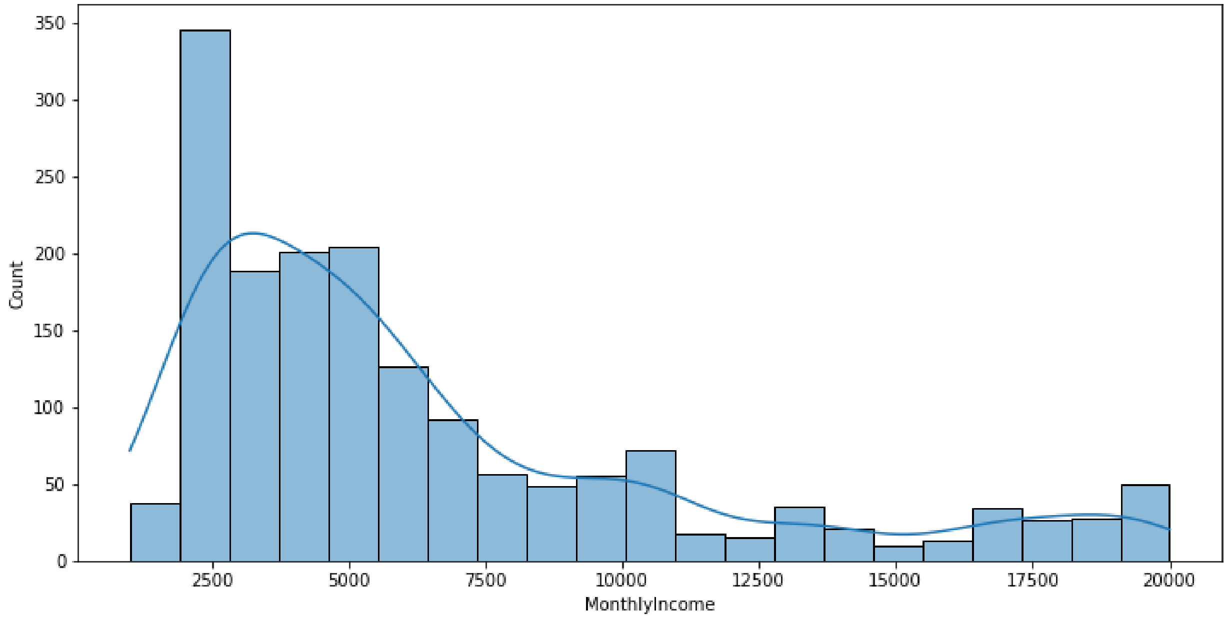

Count
EnvironmentSatisfaction
0
100
200
300
400
500
1.0
1.5
2.0
2.5
3.0
3.5
4.0

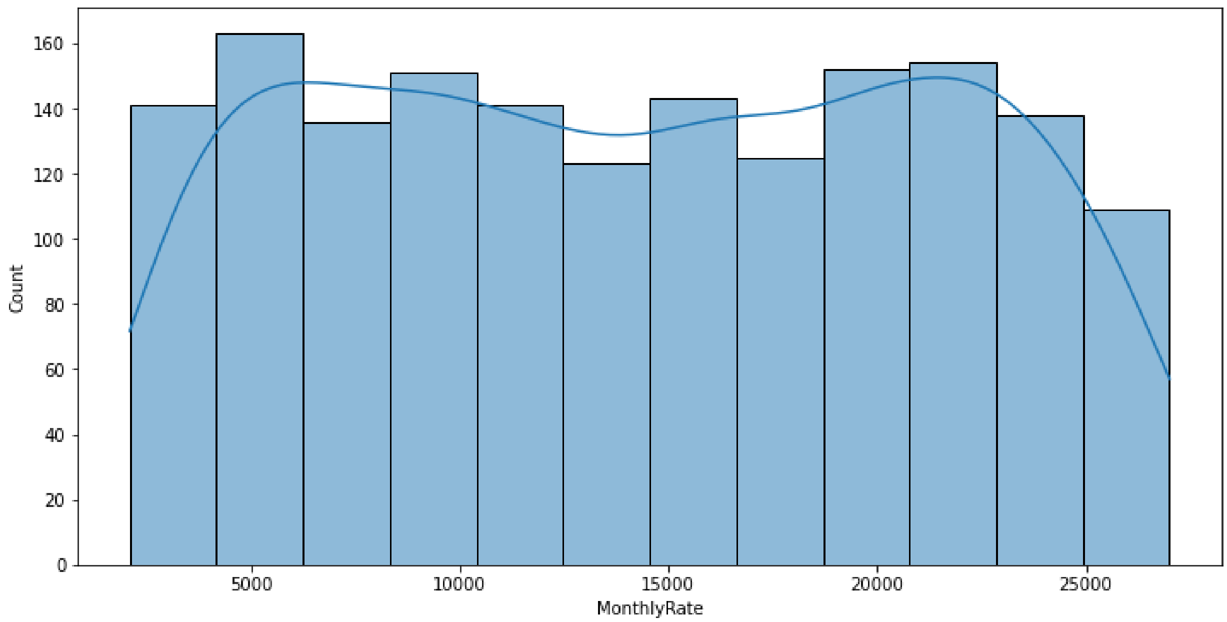

Count
HourlyRate
0
20
40
60
80
100
120
140
160
30
40
50
60
70
80
90
100

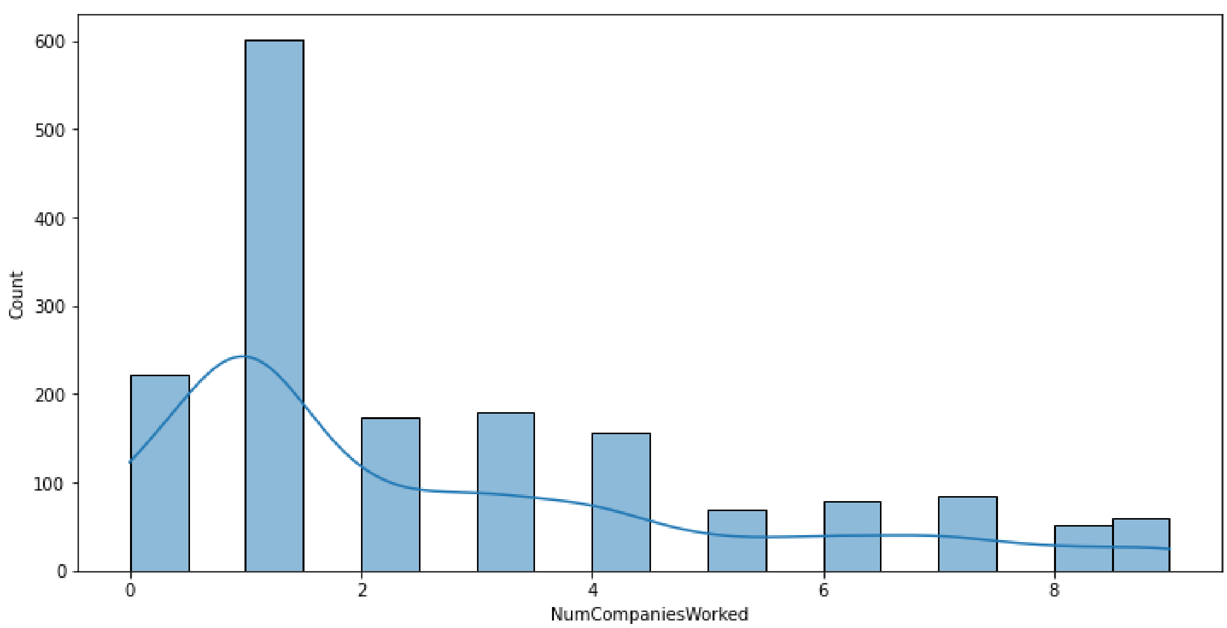

1000
800
600
400
200
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
JobInvolvement

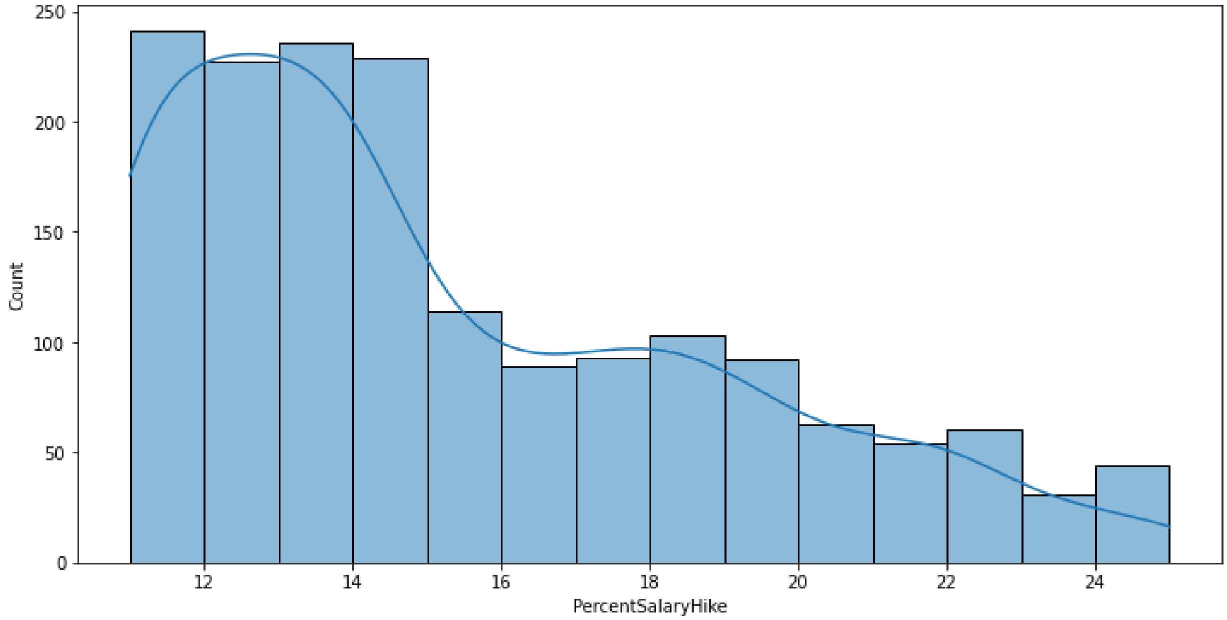

600
500
400
300
200
100
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
JobLevel

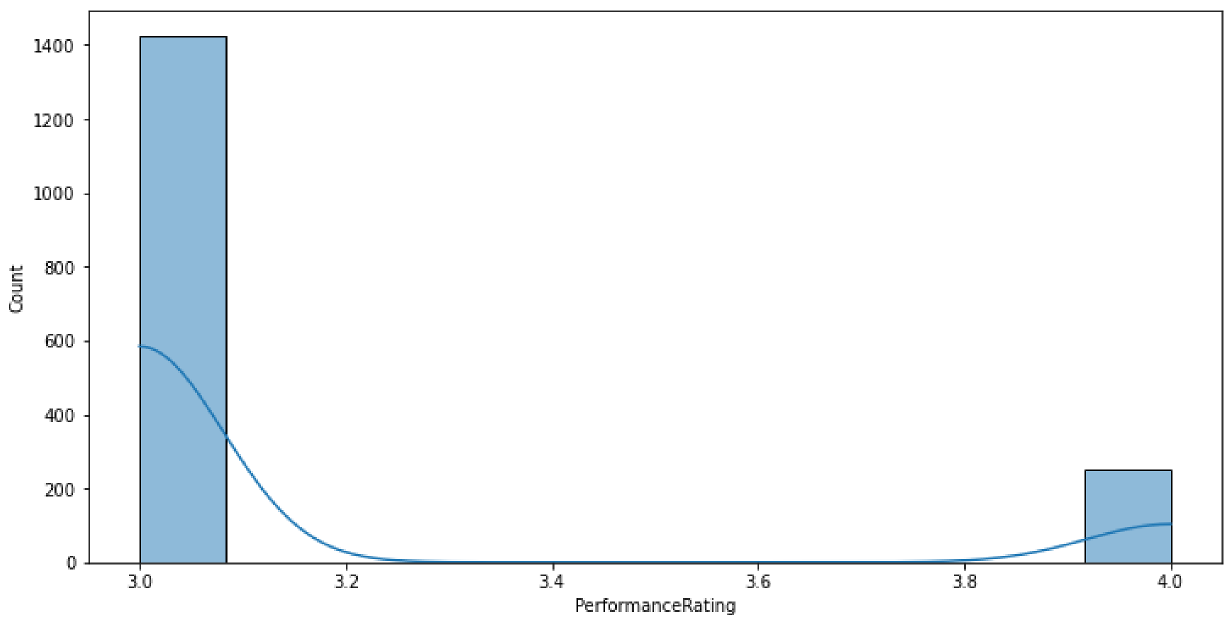

500
400
300
200
100
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
JobSatisfaction

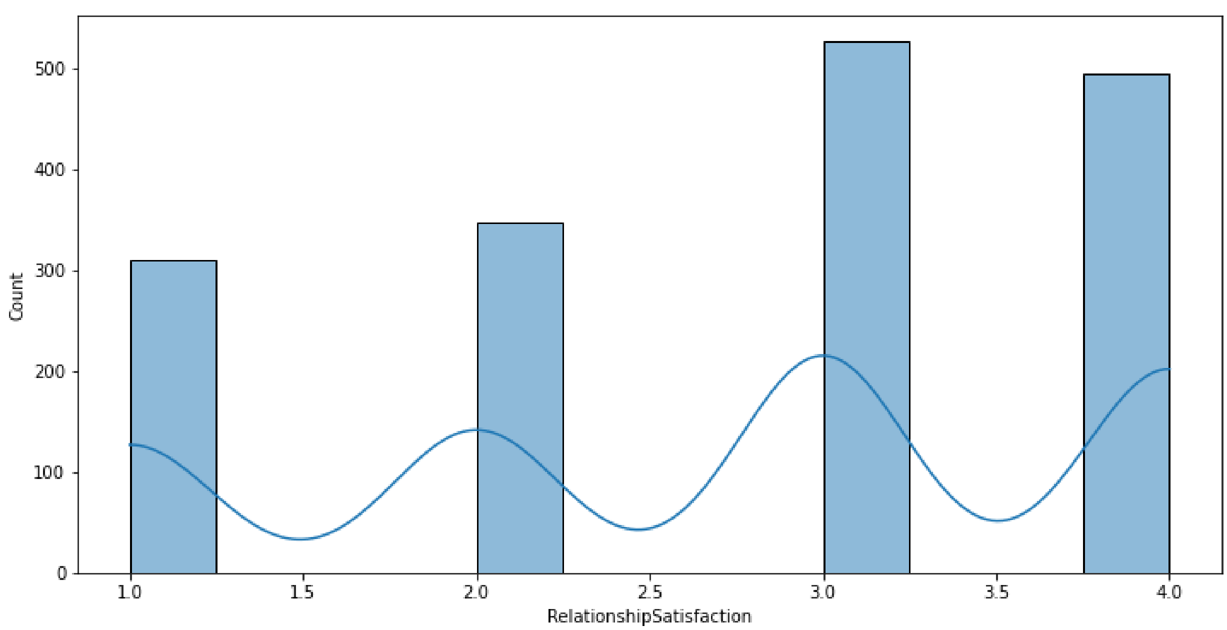

Count
MonthlyIncome
0
50
100
150
200
250
300
350
2500
5000
7500
10000
12500
15000
17500
20000

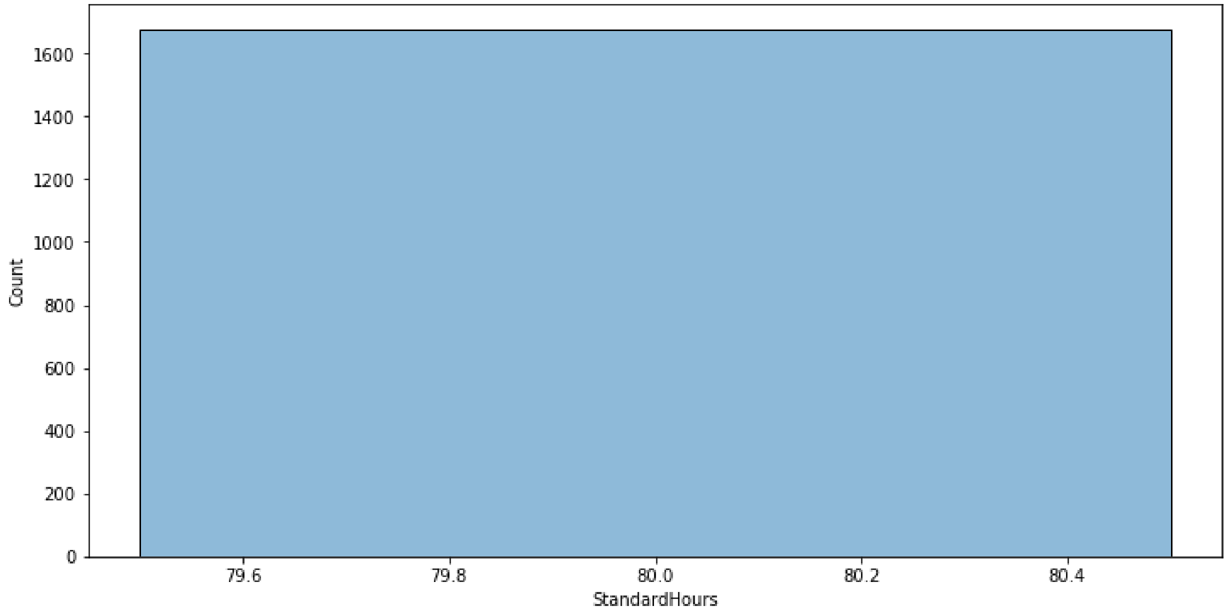

Count
MonthlyRate
0
20
40
60
80
100
120
140
160
5000
10000
15000
20000
25000

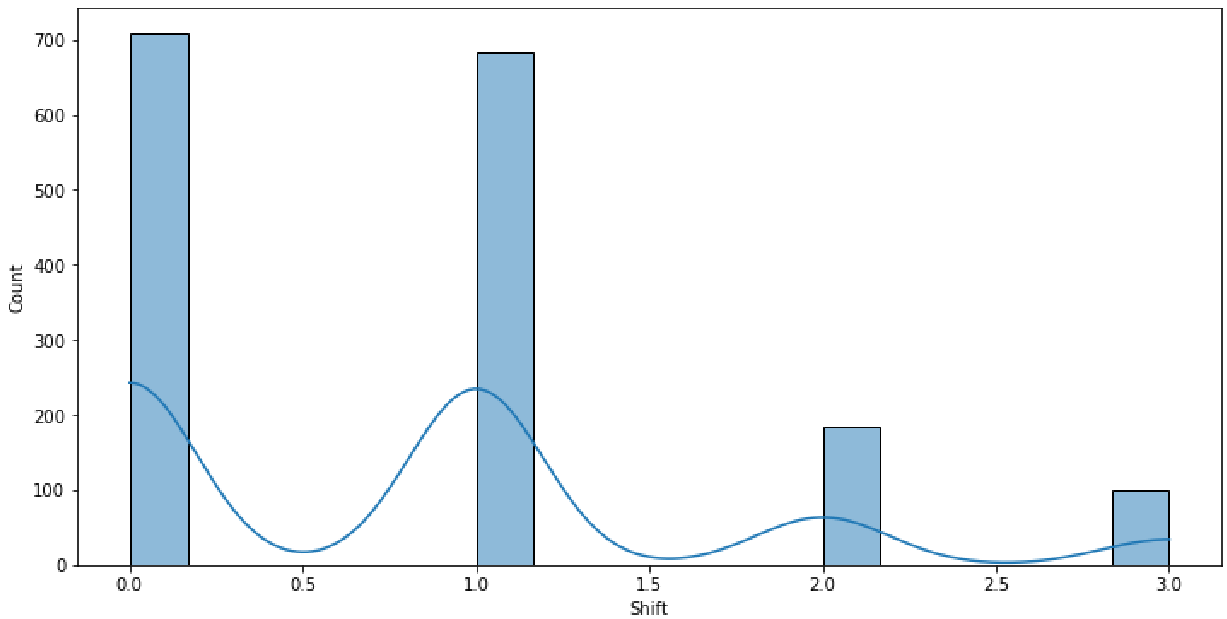

Count
NumCompaniesWorked
0
100
200
300
400
500
600
0
2
4
6
8

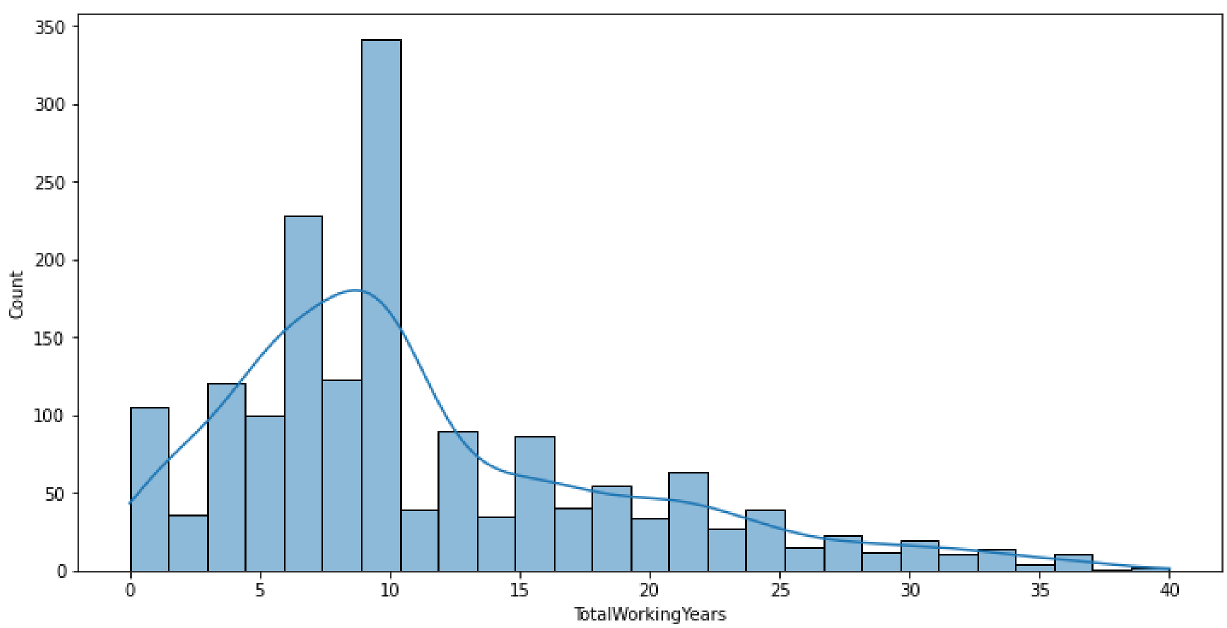

Count
PercentSalaryHike
0
50
100
150
200
250
12
14
16
18
20
22
24

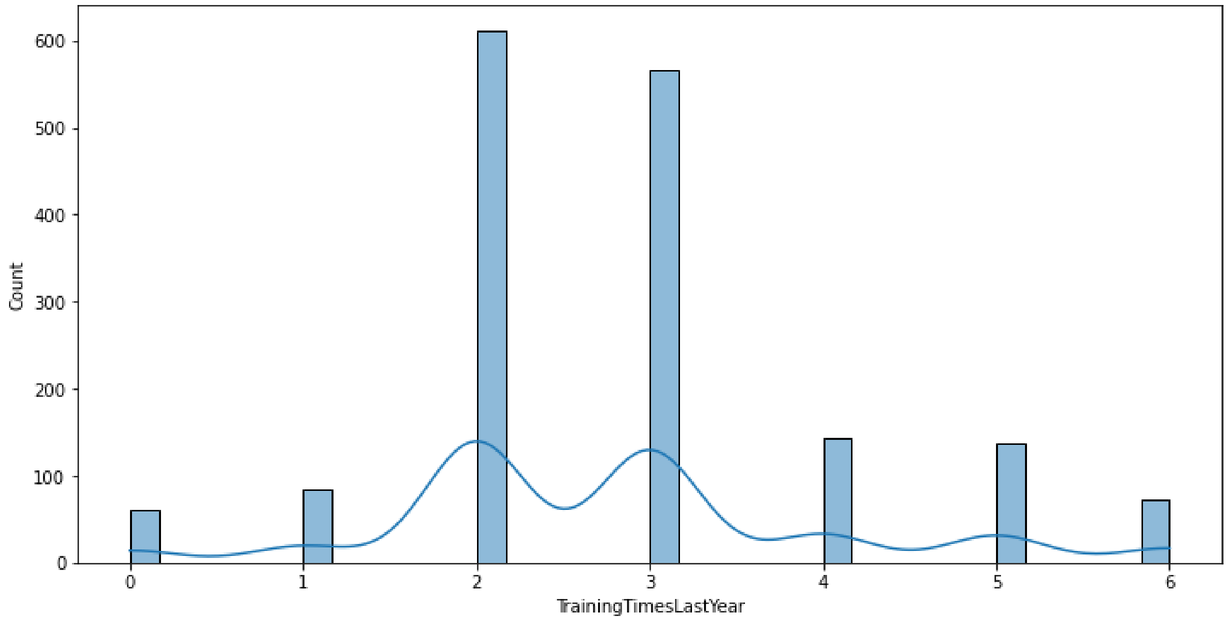

Count
PerformanceRating
0
200
400
600
800
1000
1200
1400
3.0
3.2
3.4
3.6
3.8
4.0

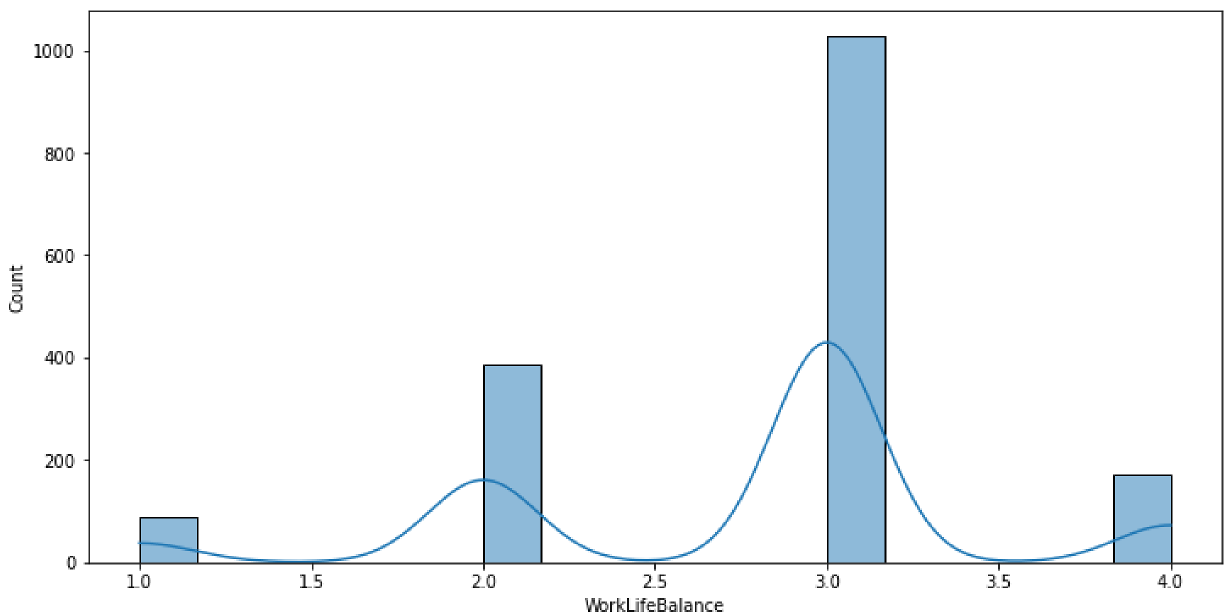

Count
RelationshipSatisfaction
0
100
200
300
400
500
1.0
1.5
2.0
2.5
3.0
3.5
4.0

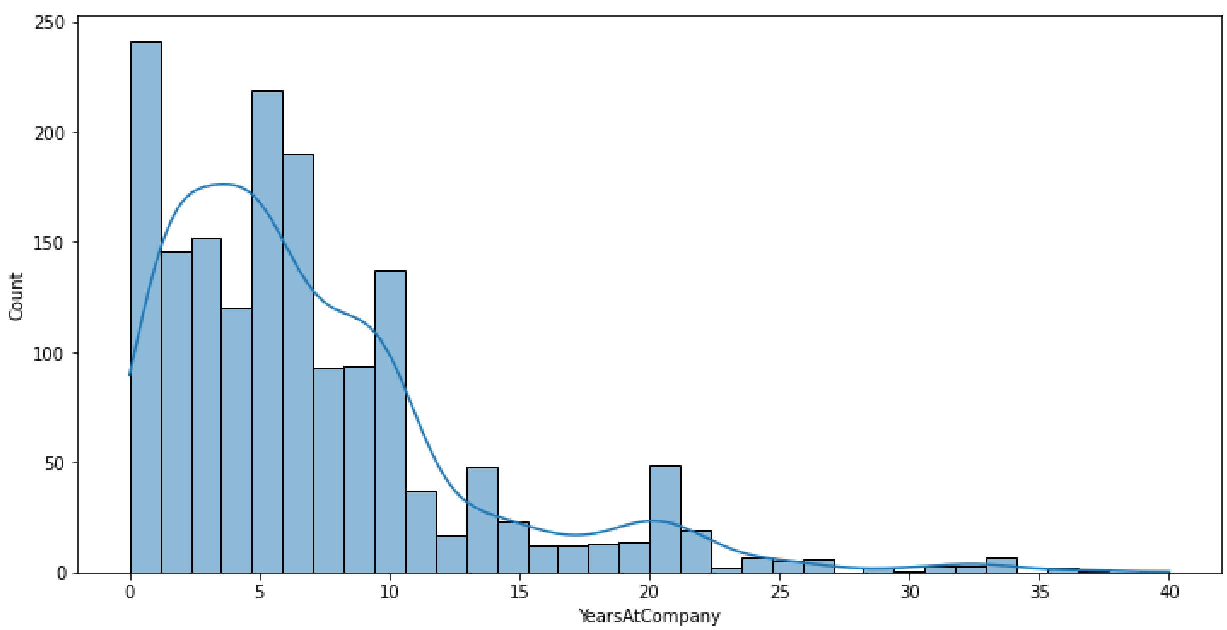

Count
StandardHours
0
200
400
600
800
1000
1200
1400
1600
79.6
79.8
80.0
80.2
80.4

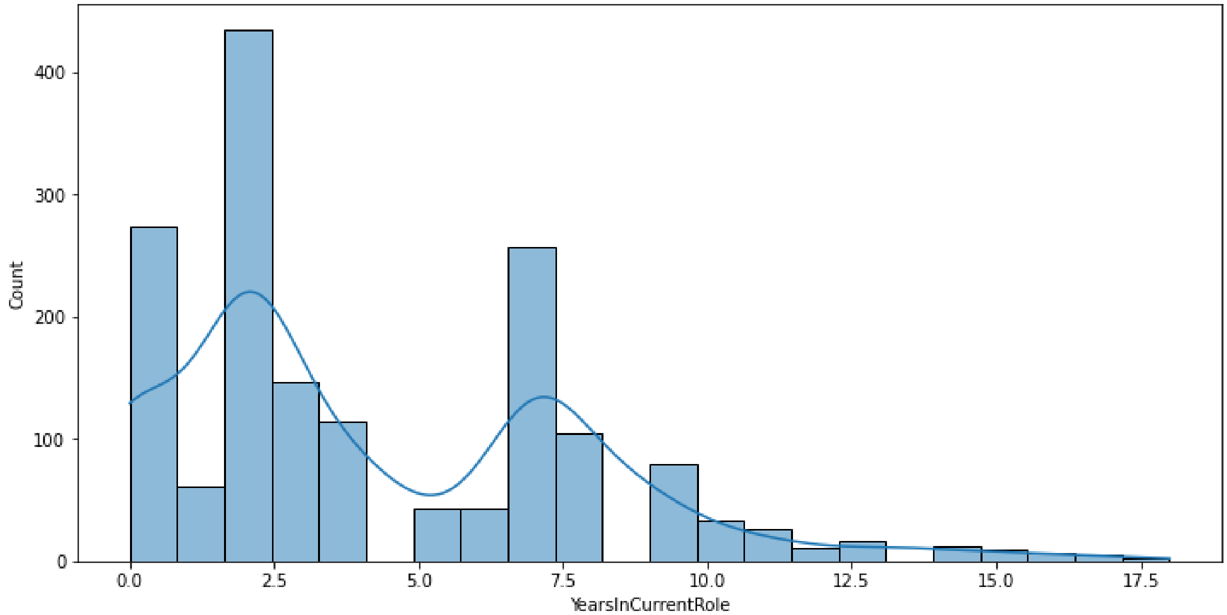

Count
Shift
0
100
200
300
400
500
600
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0

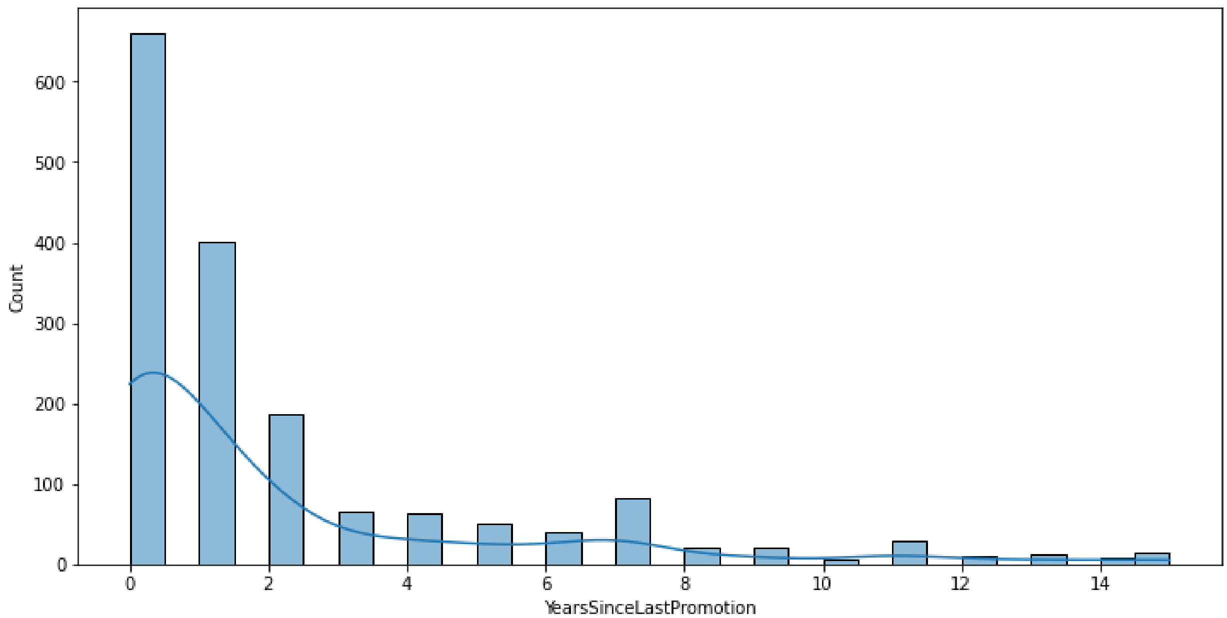

Count
TotalWorkingYears
0
50
100
150
200
250
300
350
0
5
10
15
20
25
30
35
40

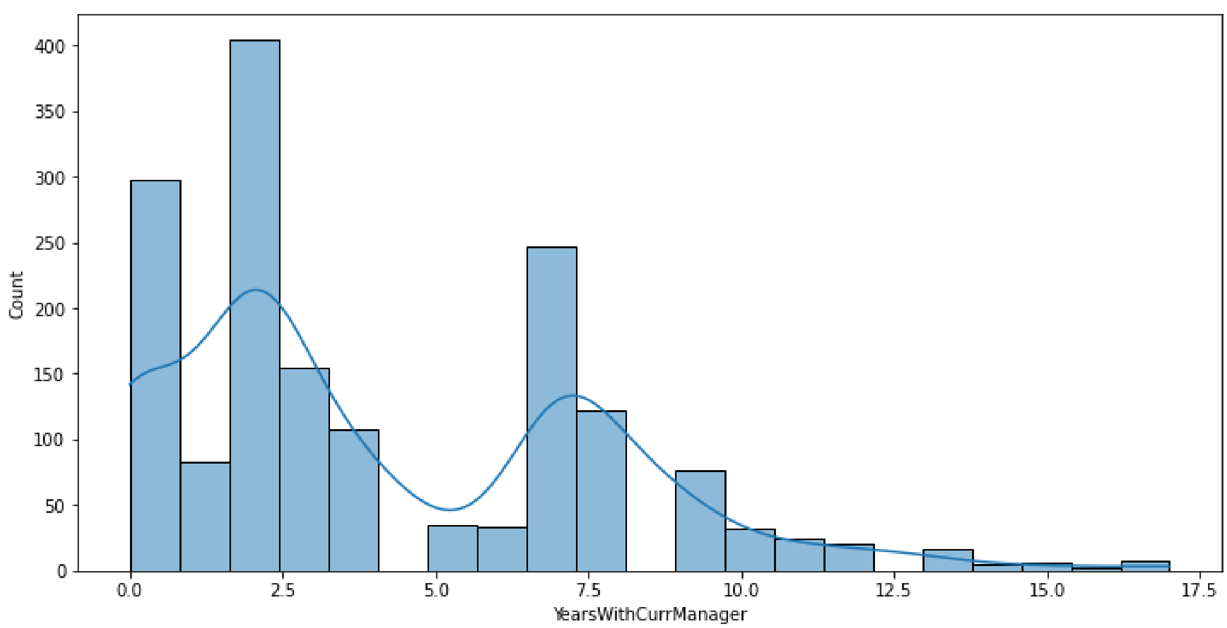

Count
TrainingTimesLastYear
0
100
200
300
400
500
600
0
1
2
3
4
5
6


Count
WorkLifeBalance
0
200
400
600
800
1000
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Count
YearsAtCompany
0
50
100
150
200
250
0
5
10
15
20
25
30
35
40


Count
YearsInCurrentRole
0
100
200
300
400
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Count
YearsSinceLastPromotion
0
100
200
300
400
500
600
0
2
4
6
8
10
12
14


Count
YearsWithCurrManager
0
50
100
150
200
250
300
350
400
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Milestone 3

In Milestone 3, you will begin the process of selecting, building, and evaluating a model. You are required to train and evaluate at least one model in this milestone. Write step-by-step for performing each of these steps. You can use any methods/tools you think are most appropriate, but you should explain/justify why you are selecting the model(s) and evaluation metric(s) you choose. It is important to think about what type of model and metric makes sense for your problem. Again, do what makes the most sense for your project. Write a short overview/conclusion of the insights gained from your model building/evaluation.

It is important to note that these milestones are meant to keep you on track for the final project submission. At any point, you can pivot or modify your project as needed based on what you discover. These milestones are not final versions; they are drafts of the many steps you need to complete along the way.

As a reminder, Teams is a great place to discuss your project with your peers. Feel free to solicit feedback/input (without creating a group project!) and collaborate on your projects with your peers.

Each milestone will build on top of each other, so make sure you do not fall behind. Submit Milestones 1-3 together. I recommend building your project milestones in a Jupyter Notebook, building upon one another. However, make sure it is clear where each milestone begins and ends.

Predict the model Accuracy

In [ ]:
```
from sklearn.linear_model import LogisticRegression
```

In [ ]:
```
# Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=.25,
                                                    random_state=42,
                                                    )
```

In [ ]:
```
logreg = LogisticRegression(penalty='l2',solver='liblinear', max_iter=250)
logreg.fit(X_train, y_train)
```

Out[ ]:
```
LogisticRegression(max_iter=250, solver='liblinear')
```

In [ ]:
```
y_pred = logreg.predict(X_test)
print("Model Accuracy:",round(logreg.score(X_test, y_test) * 100 , 0),"%")
```

```
Model Accuracy: 91.0 %
```

In [ ]:
```
## Display all the columns and rows so they can all be seen
pd.options.display.max_columns = None
pd.options.display.max_rows = None

## create a correlation matrix of the auto df and call is correlation_mat
correlation_matrix = X_train.corr()
correlation_matrix
```

Out[ ]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfac
Age	1.00	-0.01	-0.02	0.21	
DailyRate	-0.01	1.00	-0.02	-0.03	
DistanceFromHome	-0.02	-0.02	1.00	0.03	-
Education	0.21	-0.03	0.03	1.00	-
EnvironmentSatisfaction	0.02	0.01	-0.02	-0.02	
HourlyRate	0.02	0.04	0.04	-0.00	-
JobInvolvement	0.05	0.04	0.01	0.05	
JobLevel	0.50	-0.01	-0.01	0.09	
JobSatisfaction	-0.01	0.03	-0.01	0.02	-
MonthlyIncome	0.50	-0.00	-0.02	0.09	
MonthlyRate	0.03	0.01	0.06	-0.02	
NumCompaniesWorked	0.27	0.02	-0.04	0.11	
PercentSalaryHike	0.04	0.02	0.04	-0.01	-
PerformanceRating	0.02	0.02	0.04	-0.01	-
RelationshipSatisfaction	0.04	-0.01	0.01	-0.00	-
Shift	0.03	0.06	0.01	-0.01	-
TotalWorkingYears	0.68	-0.02	-0.00	0.14	
TrainingTimesLastYear	-0.04	-0.02	-0.06	0.02	-
WorkLifeBalance	0.01	-0.02	-0.02	0.00	
YearsAtCompany	0.32	-0.04	0.01	0.05	
YearsInCurrentRole	0.22	0.01	0.01	0.05	-
YearsSinceLastPromotion	0.21	-0.04	0.01	0.04	
YearsWithCurrManager	0.22	-0.05	0.01	0.05	
BusinessTravel_Travel_Frequently	-0.04	-0.02	-0.01	0.00	-
BusinessTravel_Travel_Rarely	0.04	-0.00	-0.02	-0.01	
Department_Maternity	-0.04	0.02	0.08	-0.02	-
Department_Neurology	0.10	0.01	-0.10	0.02	
EducationField_Life Sciences	-0.03	-0.02	-0.03	0.01	-
EducationField_Marketing	0.06	-0.05	0.04	0.05	
EducationField_Medical	0.01	0.04	0.01	-0.09	-
EducationField_Other	-0.03	0.02	-0.01	0.06	
EducationField_Technical Degree	-0.02	0.04	-0.00	-0.01	
Gender_Male	-0.03	0.00	0.01	-0.04	-

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfac
JobRole_Administrative	0.25	-0.04	0.01	0.09	
JobRole_Nurse	-0.09	-0.01	0.04	0.01	
JobRole_Other	-0.10	0.04	-0.03	-0.04	-
JobRole_Therapist	0.07	-0.01	-0.01	-0.03	
MaritalStatus_Married	0.10	0.02	0.01	0.01	-
MaritalStatus_Single	-0.11	-0.09	-0.01	0.01	
OverTime_Yes	0.04	0.03	0.02	-0.04	

In [ ]:
```
X.describe()
```

Out[ ]:

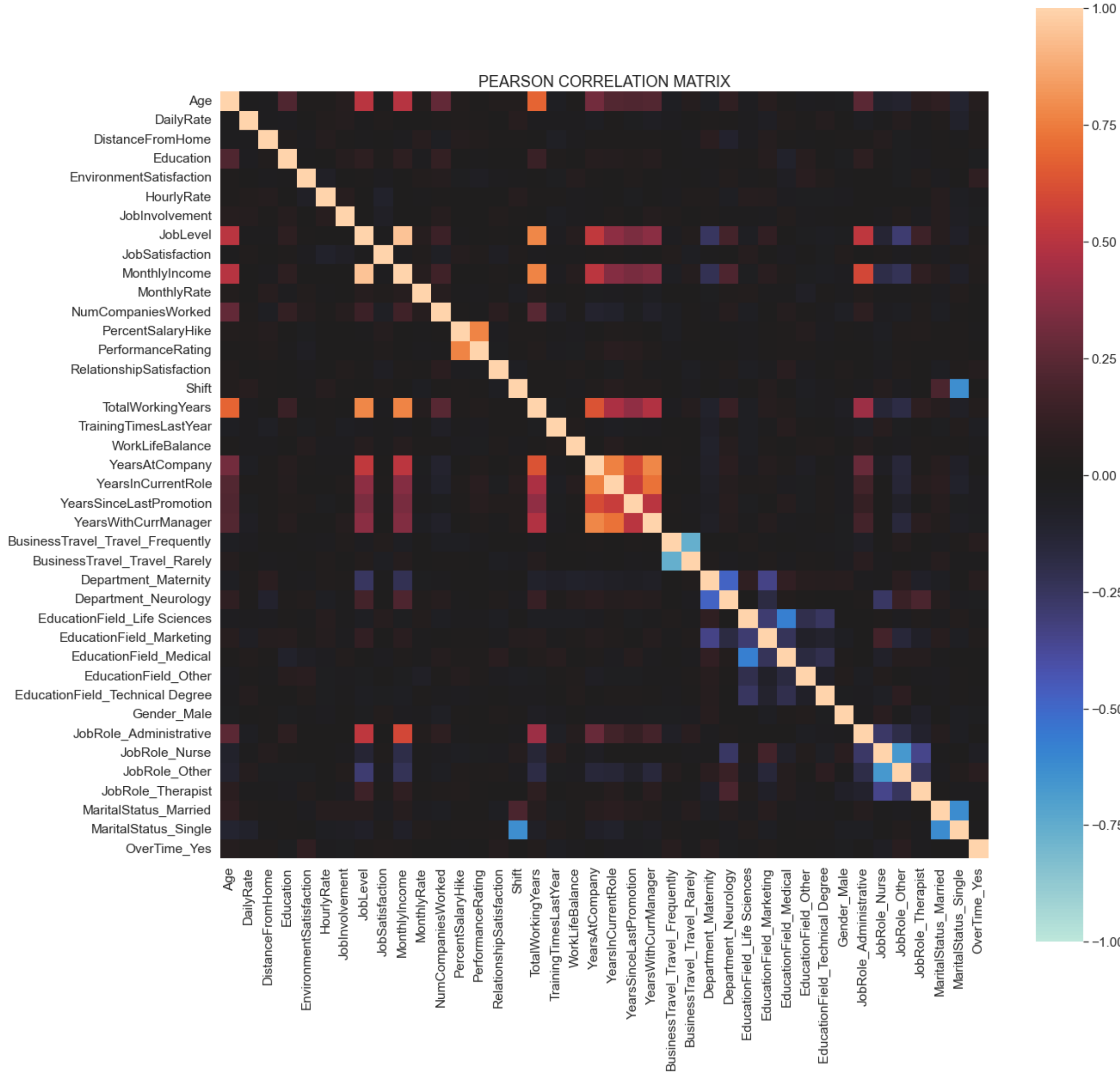
	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobIı
count	1676.00	1676.00	1676.00	1676.00	1676.00	1676.00	
mean	36.87	800.56	9.22	2.91	2.71	65.47	
std	9.13	401.59	8.16	1.03	1.10	20.21	
min	18.00	102.00	1.00	1.00	1.00	30.00	
25%	30.00	465.00	2.00	2.00	2.00	48.00	
50%	36.00	796.50	7.00	3.00	3.00	65.50	
75%	43.00	1157.00	14.00	4.00	4.00	83.00	
max	60.00	1499.00	29.00	5.00	4.00	100.00	

In [ ]:
```
pd.options.display.float_format = '{:.2f}'.format
y.describe()
```

Out[ ]:
```
count   1676.00
mean       0.12
std        0.32
min        0.00
25%        0.00
50%        0.00
75%        0.00
max        1.00
Name: Attrition, dtype: float64
```

In [ ]:
```
## look for multicollinearity of features
fig, ax = plt.subplots(figsize=(20, 20))

sns.heatmap(X_train.corr(), center=0,
            vmin=-1, vmax=1,  square=True)

# title
plt.title('PEARSON CORRELATION MATRIX', fontsize=18)
```

```
plt.show()
```



In [ ]:

```
df_correlations = X_train.corr().abs().stack().reset_index().sort_values(0, ascending=

# zip the variable name columns in a new column named "pairs"
df_correlations['pairs'] = list(zip(df_correlations.level_0, df_correlations.level_1))

# set index to pairs
df_correlations.set_index(['pairs'], inplace = True)

# rename our results column to correlation
df_correlations.rename(columns={0: "correlation"}, inplace=True)

# Drop 1:1 correlations to get rid of self pairs
df_correlations.drop(df_correlations[df_correlations['correlation'] == 1.000000].index

# view pairs above 75% correlation and below 90% correlation (engineered features will
df_correlations[(df_correlations.correlation>.75) & (df_correlations.correlation<.95)]
```

Out[ ]:

	level_0	level_1	correla
pairs			
(TotalWorkingYears, JobLevel)	TotalWorkingYears	JobLevel	
(JobLevel, TotalWorkingYears)	JobLevel	TotalWorkingYears	
(YearsAtCompany, YearsWithCurrManager)	YearsAtCompany	YearsWithCurrManager	
(YearsWithCurrManager, YearsAtCompany)	YearsWithCurrManager	YearsAtCompany	
(MonthlyIncome, TotalWorkingYears)	MonthlyIncome	TotalWorkingYears	
(TotalWorkingYears, MonthlyIncome)	TotalWorkingYears	MonthlyIncome	
(PercentSalaryHike, PerformanceRating)	PercentSalaryHike	PerformanceRating	
(PerformanceRating, PercentSalaryHike)	PerformanceRating	PercentSalaryHike	
(BusinessTravel_Travel_Frequently, BusinessTravel_Travel_Rarely)	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	
(BusinessTravel_Travel_Rarely, BusinessTravel_Travel_Frequently)	BusinessTravel_Travel_Rarely	BusinessTravel_Travel_Frequently	
(YearsAtCompany, YearsInCurrentRole)	YearsAtCompany	YearsInCurrentRole	
(YearsInCurrentRole, YearsAtCompany)	YearsInCurrentRole	YearsAtCompany	

In [ ]:

```
df_correlations = X_train.corr().abs().stack().reset_index().sort_values(0, ascending=
df_correlations.loc[df_correlations['level_0'] == 'TotalWorkingYears'].sort_values(0,
```

Out[ ]:

	level_0	level_1	0
656	TotalWorkingYears	TotalWorkingYears	1.00
647	TotalWorkingYears	JobLevel	0.78
649	TotalWorkingYears	MonthlyIncome	0.77
640	TotalWorkingYears	Age	0.68
659	TotalWorkingYears	YearsAtCompany	0.63
662	TotalWorkingYears	YearsWithCurrManager	0.48
660	TotalWorkingYears	YearsInCurrentRole	0.47
673	TotalWorkingYears	JobRole_Administrative	0.43
661	TotalWorkingYears	YearsSinceLastPromotion	0.39
651	TotalWorkingYears	NumCompaniesWorked	0.24
675	TotalWorkingYears	JobRole_Other	0.18
643	TotalWorkingYears	Education	0.14
674	TotalWorkingYears	JobRole_Nurse	0.13
666	TotalWorkingYears	Department_Neurology	0.13
665	TotalWorkingYears	Department_Maternity	0.09
676	TotalWorkingYears	JobRole_Therapist	0.09
678	TotalWorkingYears	MaritalStatus_Single	0.07
668	TotalWorkingYears	EducationField_Marketing	0.05
672	TotalWorkingYears	Gender_Male	0.05
677	TotalWorkingYears	MaritalStatus_Married	0.05
650	TotalWorkingYears	MonthlyRate	0.05
664	TotalWorkingYears	BusinessTravel_Travel_Rarely	0.04
667	TotalWorkingYears	EducationField_Life Sciences	0.04
657	TotalWorkingYears	TrainingTimesLastYear	0.04
670	TotalWorkingYears	EducationField_Other	0.04
679	TotalWorkingYears	OverTime_Yes	0.03
653	TotalWorkingYears	PerformanceRating	0.03
669	TotalWorkingYears	EducationField_Medical	0.03
663	TotalWorkingYears	BusinessTravel_Travel_Frequently	0.03
671	TotalWorkingYears	EducationField_Technical Degree	0.02
641	TotalWorkingYears	DailyRate	0.02
648	TotalWorkingYears	JobSatisfaction	0.02
652	TotalWorkingYears	PercentSalaryHike	0.02

	level_0	level_1	0
658	TotalWorkingYears	WorkLifeBalance	0.01
645	TotalWorkingYears	HourlyRate	0.01
654	TotalWorkingYears	RelationshipSatisfaction	0.01
655	TotalWorkingYears	Shift	0.01
644	TotalWorkingYears	EnvironmentSatisfaction	0.01
646	TotalWorkingYears	JobInvolvement	0.00
642	TotalWorkingYears	DistanceFromHome	0.00

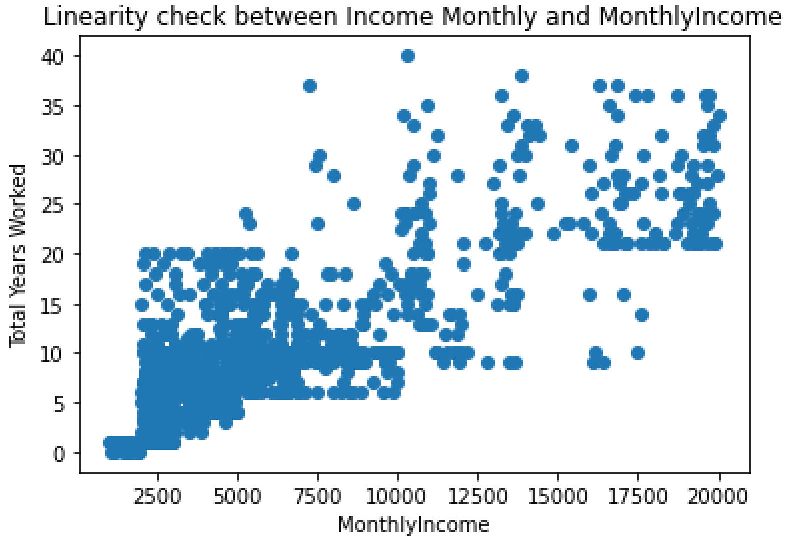
Observation:

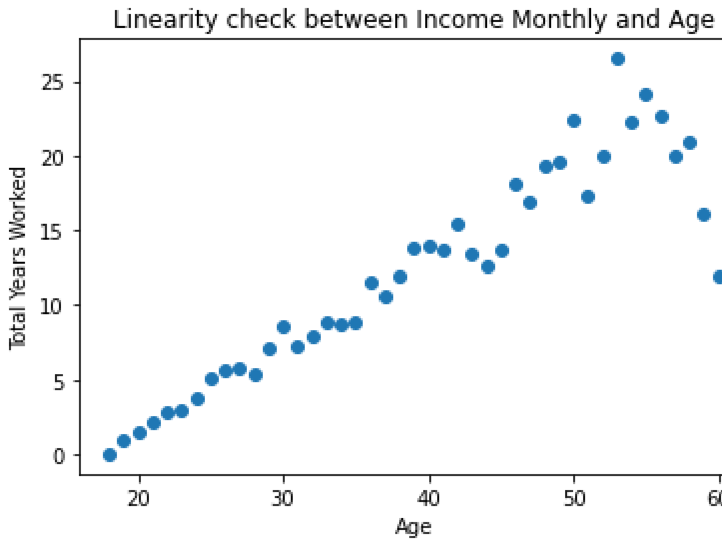
Total Years worked has a high positive corelation with the following:

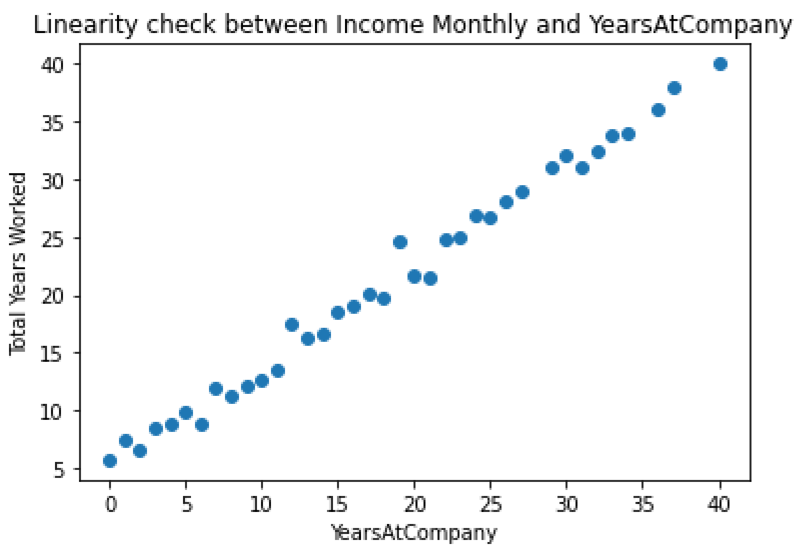
- Job Level
- Monthly Income
- Years At Company

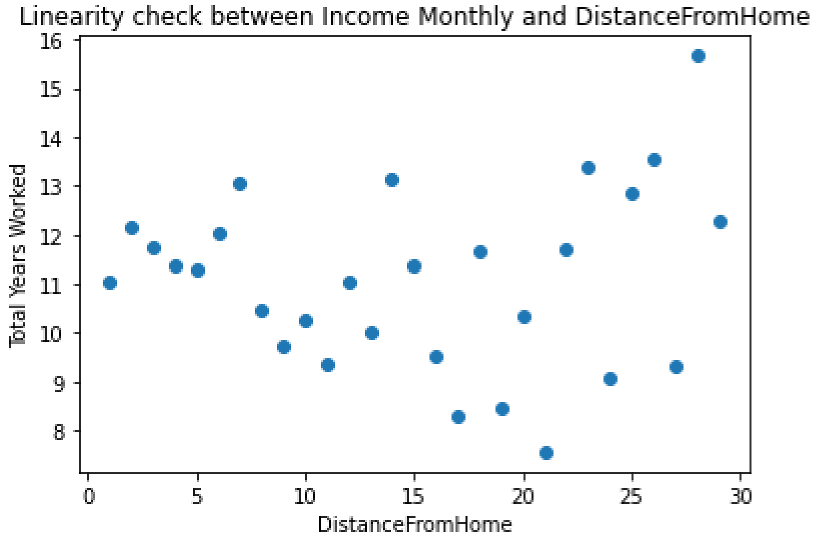
This means that for the number of years worked within the a company, there is a high likelihood these individuals are paid well, and have spent a long while in a prestigious career within their company and are less likely to leave.

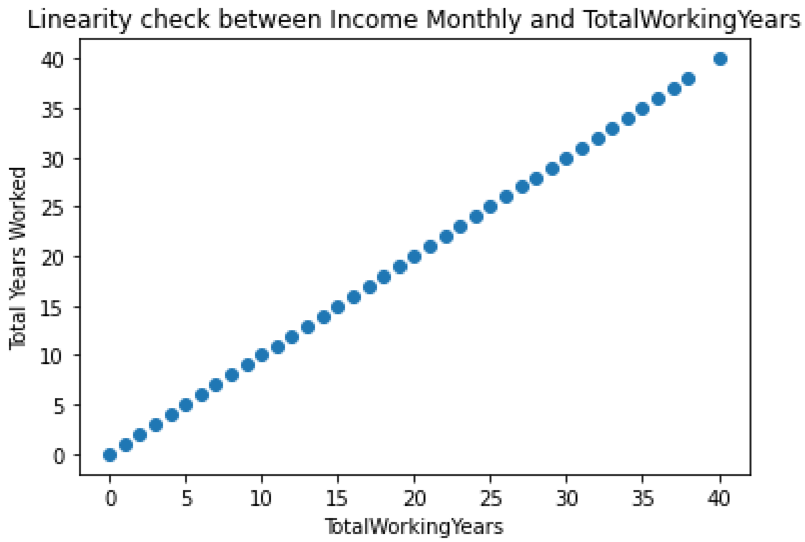
In [ ]:
```
## Creating linear chart between TotalYearsWorked and couple of constant variables ava
## Among various features available in the dataset,
## I have chosen below constant variables for plotting which are most useful compared
linearCols = [ 'MonthlyIncome', 'Age', 'YearsAtCompany', 'DistanceFromHome', 'TotalWor

## Plotting linear graph between Price and vriables present in the above list
for col in linearCols:
    lat_changes = X_train.groupby(col)['TotalWorkingYears'].mean()
    plt.scatter(lat_changes.index, lat_changes)
    plt.title("Linearity check between Income Monthly and "+ col)
    plt.xlabel(col)
    plt.ylabel('Total Years Worked')
    plt.show()
```

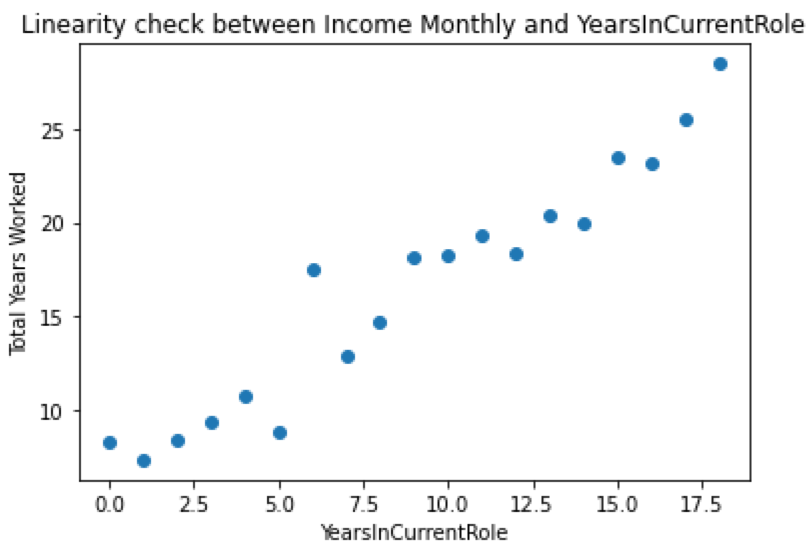

Linearity check between Income Monthly and MonthlyIncome
Total Years Worked
MonthlyIncome


Linearity check between Income Monthly and Age
Total Years Worked
Age


Linearity check between Income Monthly and YearsAtCompany
Total Years Worked
YearsAtCompany







Observation:

- MonthlyIncome vs TotalWorkingYears: Total years worked vs monthly income was linear in value as even the lowest of incomes still increased more than the previous year worked.

- Age vs TotalWorkingYears: We see something interesting at the age of 50-55 where the total years worked vs age significantly drops indicating that the median age for people in this dataset leaving the workforce around 55 while outliers do exist that make it past the working age of 60.

- YearsAtCompany vs TotalWorkingYears: The longer a person has total working years, it would seem that the is a linear climb of individuals working with the same company

- DistanceFromHome vs TotalWorkingYears: Seems somewhat linear but in two different ways, individuals who live closer work at the same place longer, and people that live farther away tend to leave sooner after 10 miles of distance, while at the same time an equal amount of people after 10 miles will continue to work for the same institution

- TotalWorkingYears vs TotalWorkingYears: 1:1 relationship so should be a straight line.

- YearsInCurrentRole vs TotalWorkingYears: Those who work the same role tend to stay in the same institution so a linear relationship is created.

RandomForest Classification

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
rd = RandomForestClassifier()
```

In [ ]:
```
rd.fit(X_train,y_train)
```

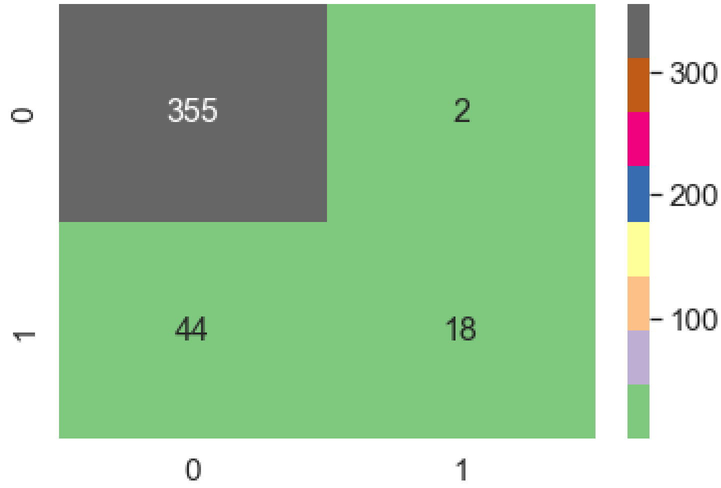
Out[ ]:
```
RandomForestClassifier()
```

In [ ]:
```
y_pred = rd.predict(X_test)
```

In [ ]:
```
print(f'RandomForest Classification Score:  {accuracy_score(y_test, y_pred)}')
```

```
RandomForest Classification Score:  0.8902147971360382
```

In [ ]:
```
cm = confusion_matrix(y_test, y_pred)
sns.set(font_scale=1.4)
sns.heatmap(cm, annot=True, annot_kws={"size": 16}, fmt="d", cmap='Accent')
plt.show()
```



Observation:

- True positives where high along with true negatives
- Larger scores of false negatives were counted over false positives

In [ ]: