IBM HR Analytics Employee Attrition & Performance

Import libraries for data analysis and wrangling

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

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
df= pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
df.head()
```

Out[ ]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely	591	Research & Development	2	1

5 rows × 35 columns

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

Out[ ]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Employ
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1
mean	36.923810	802.485714	9.192517	2.912925	1.0	1
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	1
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1
max	60.000000	1499.000000	29.000000	5.000000	1.0	2

8 rows × 26 columns

In [ ]:

```
df.describe(include='O')
```

Out[ ]:

	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalSt
count	1470	1470	1470	1470	1470	1470	
unique	2	3	3	6	2	9	
top	No	Travel_Rarely	Research & Development	Life Sciences	Male	Sales Executive	Ma
freq	1233	1043	961	606	882	326	

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

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

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

In [ ]:
```
df.nunique()
```

Out[ ]:

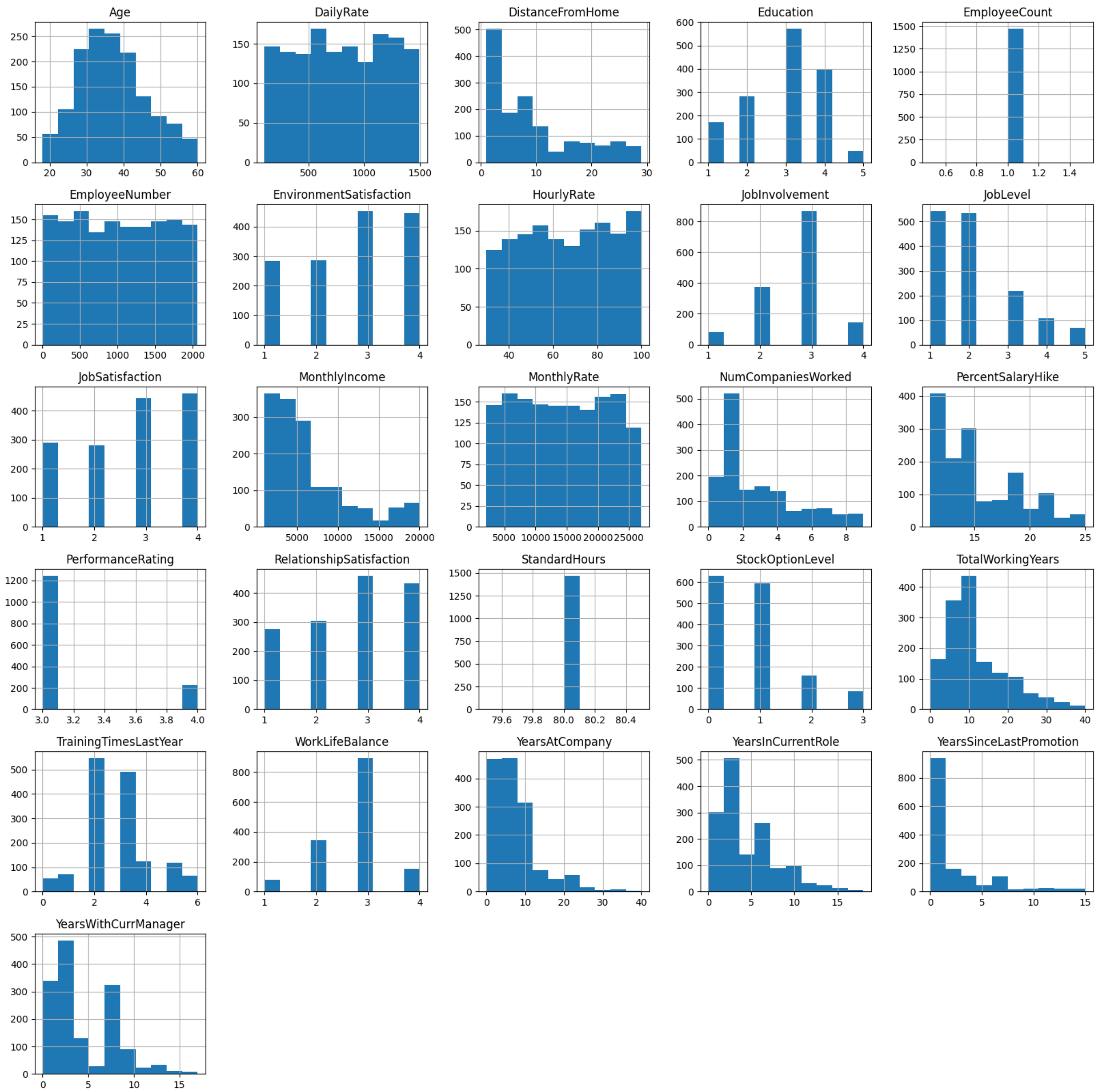
```
Age                           43
Attrition                      2
BusinessTravel                 3
DailyRate                    886
Department                     3
DistanceFromHome              29
Education                      5
EducationField                 6
EmployeeCount                  1
EmployeeNumber              1470
EnvironmentSatisfaction        4
Gender                         2
HourlyRate                    71
JobInvolvement                 4
JobLevel                       5
JobRole                        9
JobSatisfaction                4
MaritalStatus                  3
MonthlyIncome               1349
MonthlyRate                 1427
NumCompaniesWorked            10
Over18                         1
OverTime                       2
PercentSalaryHike             15
PerformanceRating              2
RelationshipSatisfaction       4
StandardHours                  1
StockOptionLevel               4
TotalWorkingYears             40
TrainingTimesLastYear          7
WorkLifeBalance                4
YearsAtCompany                37
YearsInCurrentRole            19
YearsSinceLastPromotion       16
YearsWithCurrManager          18
dtype: int64
```

At the first look we can see that we have 1470 record and 35 columns. We can also see that there are no missing values in the dataset. We have 26 columns with numerical values and 9 columns with categorical values. We will have to convert the categorical values to numerical values later on. There are some columns that we can drop because they are not relevant for our analysis or they have no variance like EmployeeCount, EmployeeNumber, Over18 and StandardHours.

Now we Start with exploration of the Attrition column.

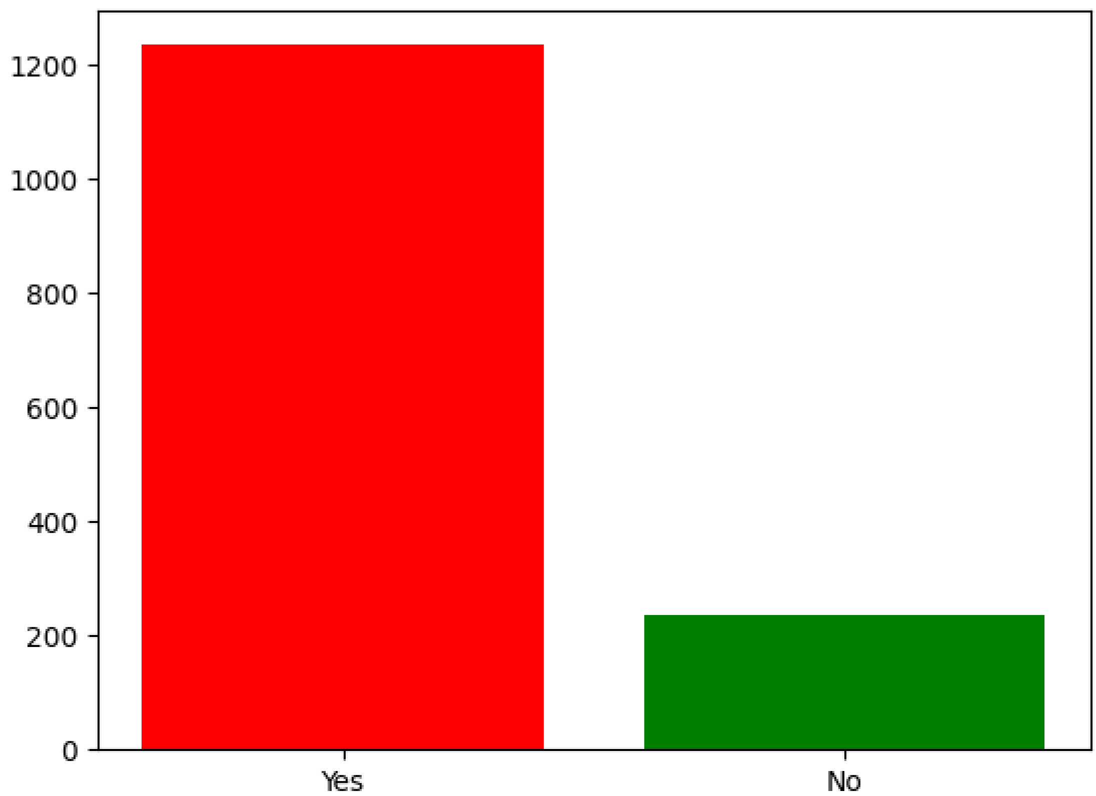
In [ ]:

```
df.hist(figsize=(20,20));
```



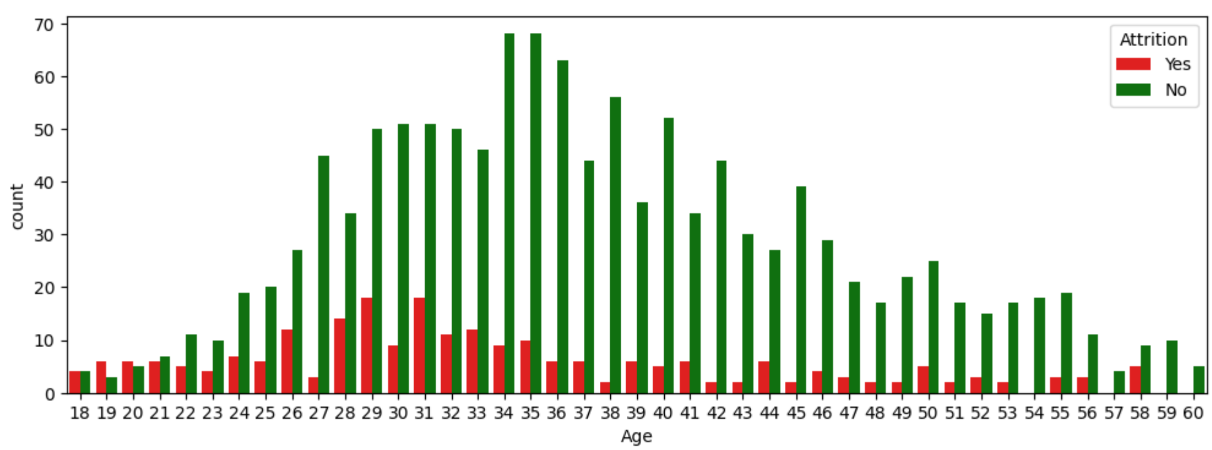
In [ ]:

```
plt.bar(df['Attrition'].unique(), df['Attrition'].value_counts(), color=['red', 'gr
```

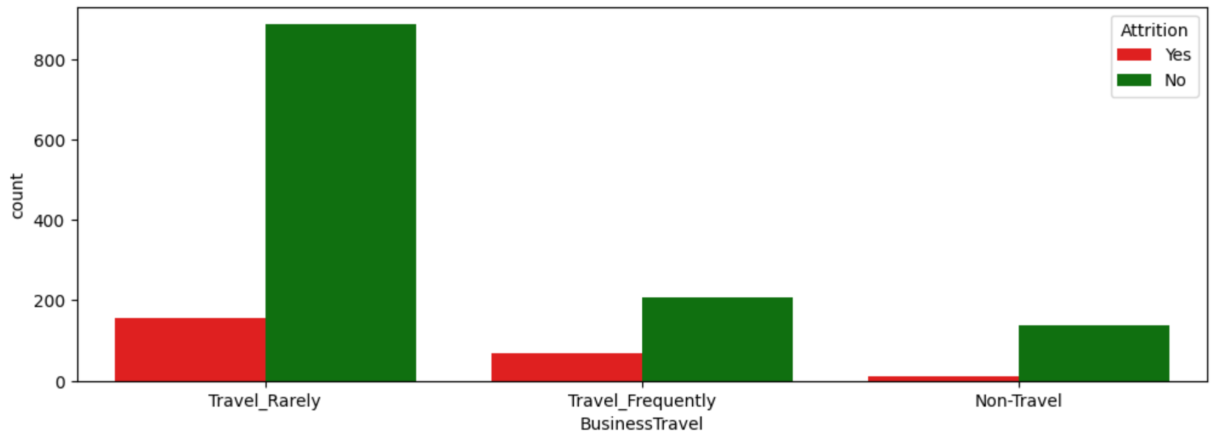


We can see that data is imbalanced. We have 1233 records with Attrition = No and 237 records with Attrition = Yes. We will have to balance the data later on.

In [ ]:
```
# Attrition by Age
plt.figure(figsize=(12,4))
sns.countplot(x='Age', hue='Attrition', data=df,hue_order=['Yes','No'],palette=['re
```

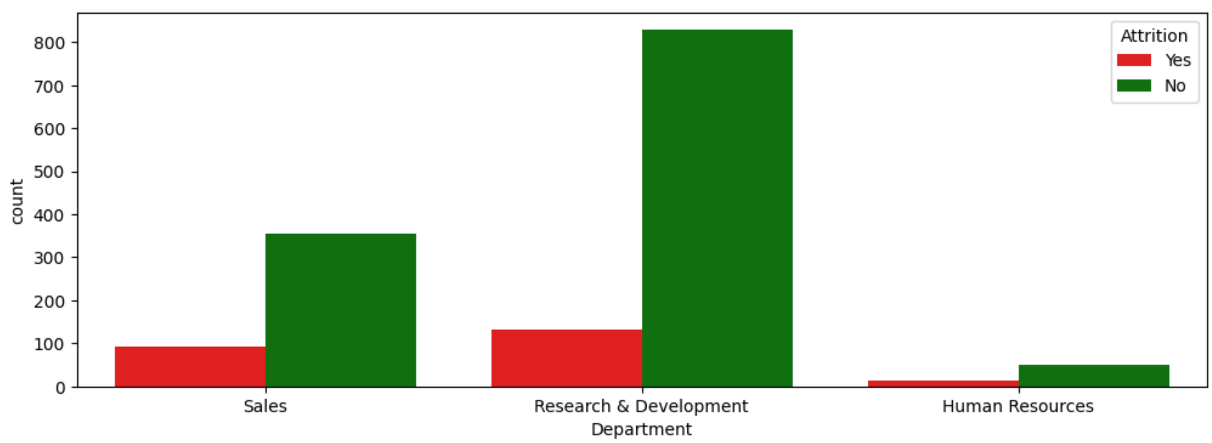


In [ ]:
```
# Attrition by Business Travel
plt.figure(figsize=(12,4))
sns.countplot(x='BusinessTravel', hue='Attrition', data=df,hue_order=['Yes','No'],p
```



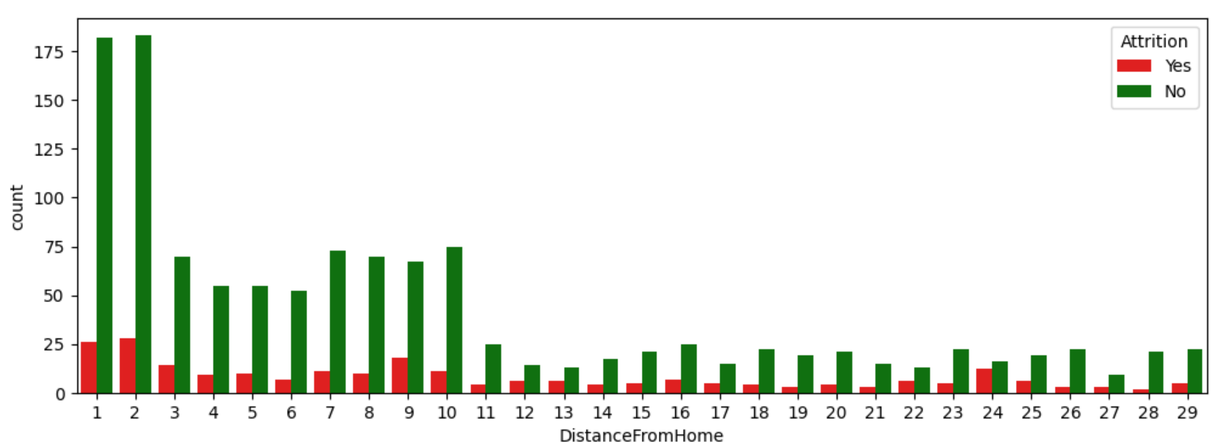
In [ ]:

```
# Attrition by Department
plt.figure(figsize=(12,4))
sns.countplot(x='Department', hue='Attrition', data=df,hue_order=['Yes','No'],palet
```



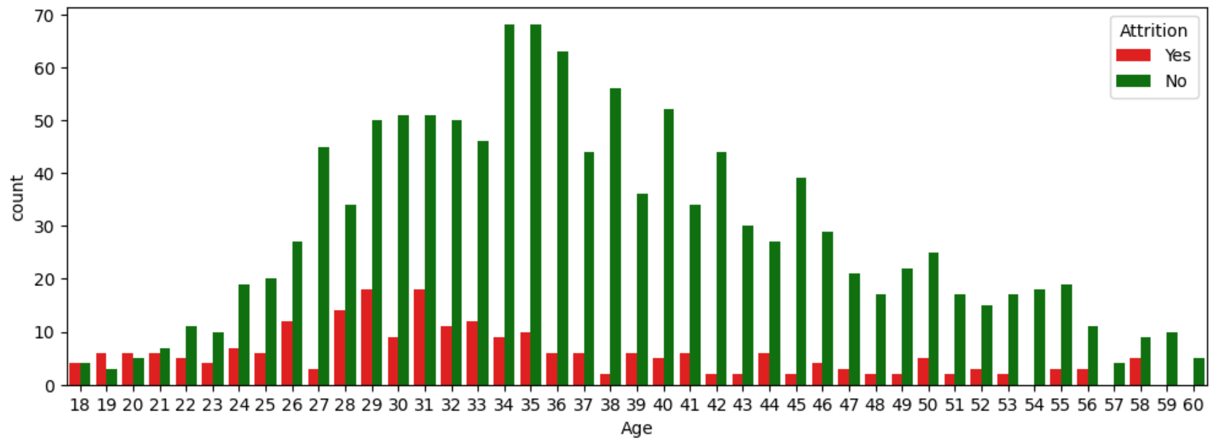
In [ ]:

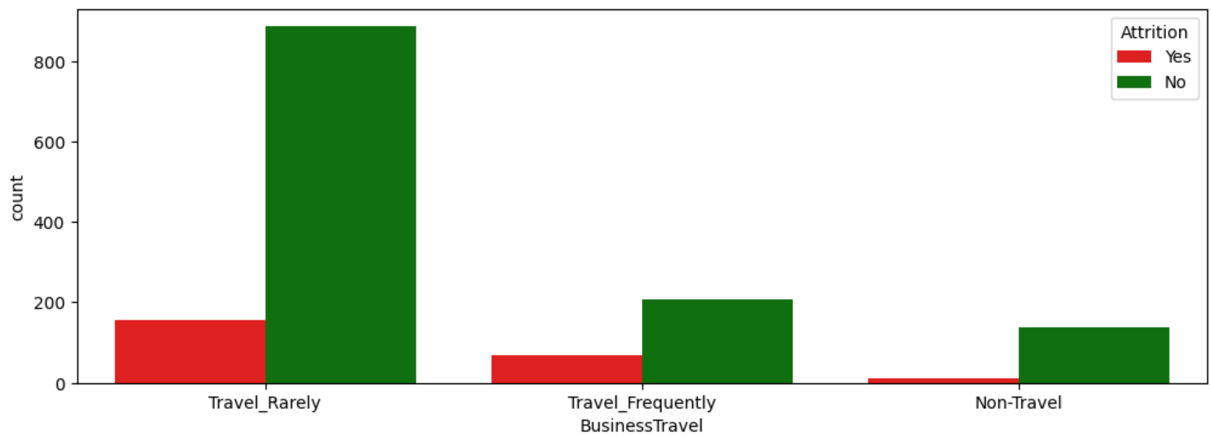
```
# Attrition by DistanceFromHome
plt.figure(figsize=(12,4))
sns.countplot(x='DistanceFromHome', hue='Attrition', data=df,hue_order=['Yes','No']
```

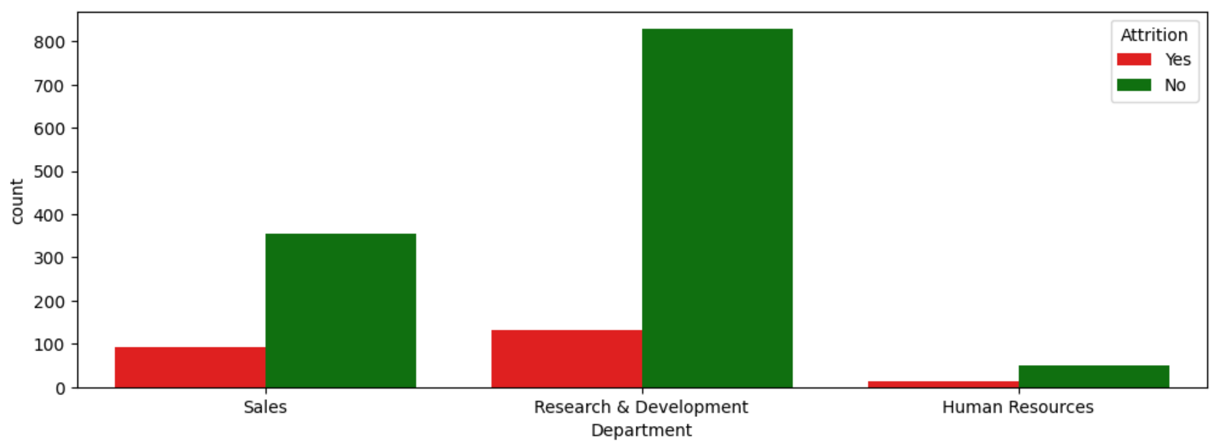


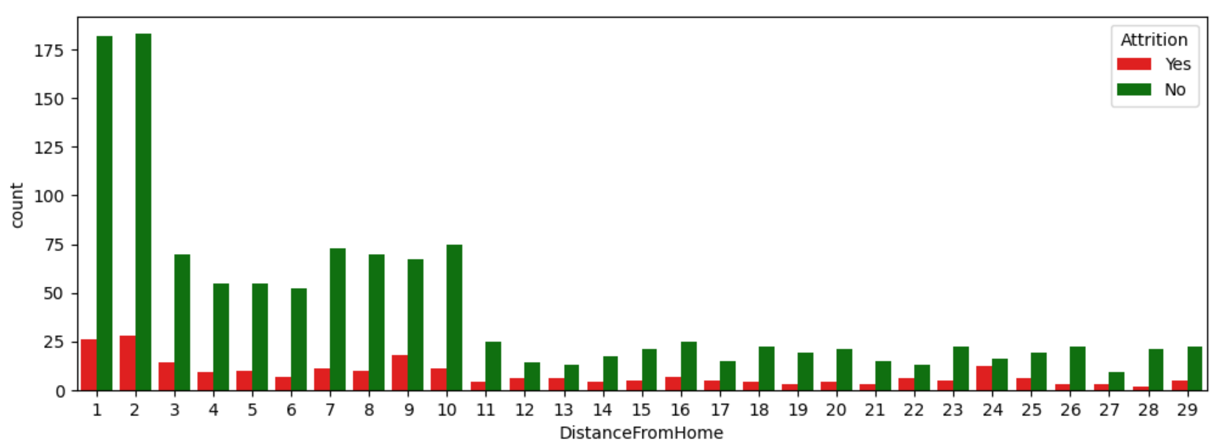
In [ ]:

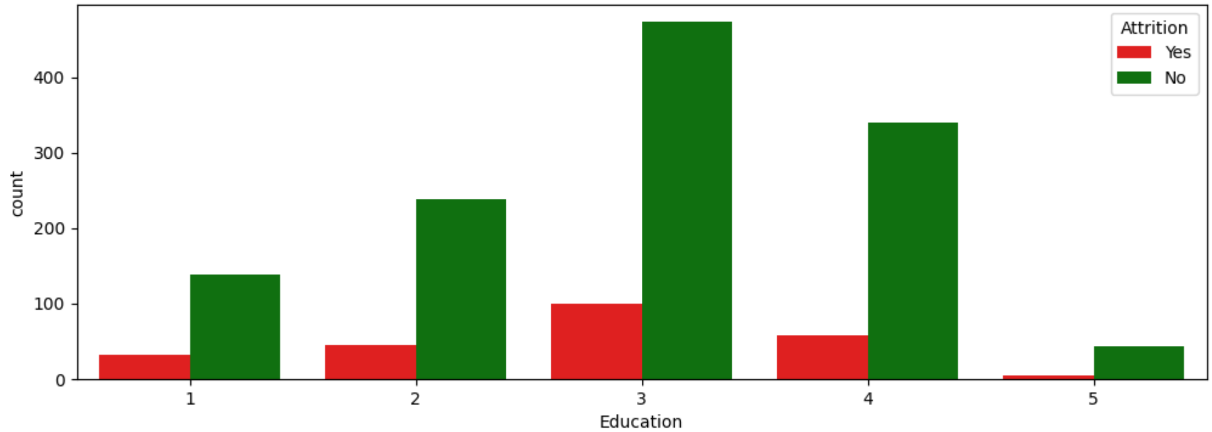
```
# Attrition for all the columns
for i, col in enumerate(df.drop(['Attrition','EmployeeCount','EmployeeNumber','Dail
    plt.figure(i,figsize=(12,4))
    sns.countplot(x=col, hue='Attrition', data=df,hue_order=['Yes','No'],palette=['
```

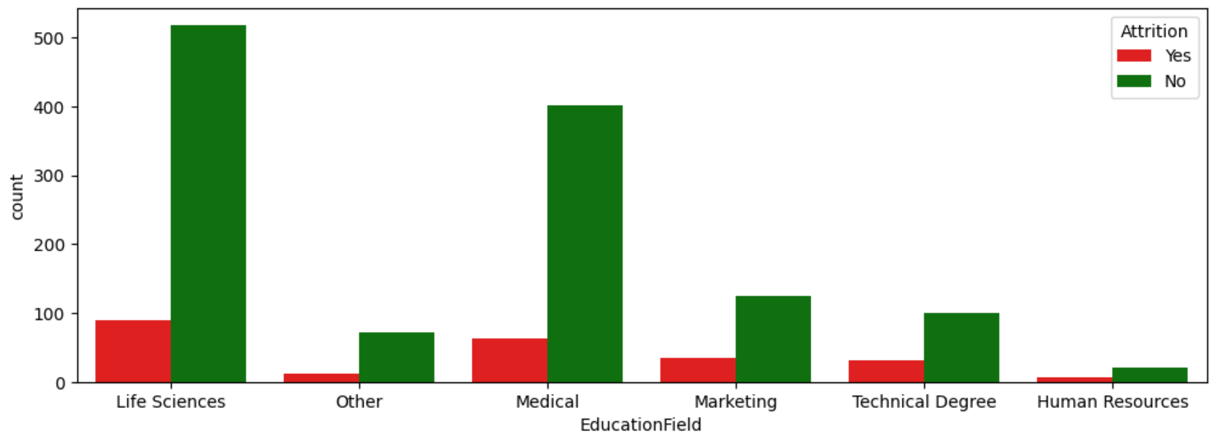

Attrition
Yes
No
count
Age

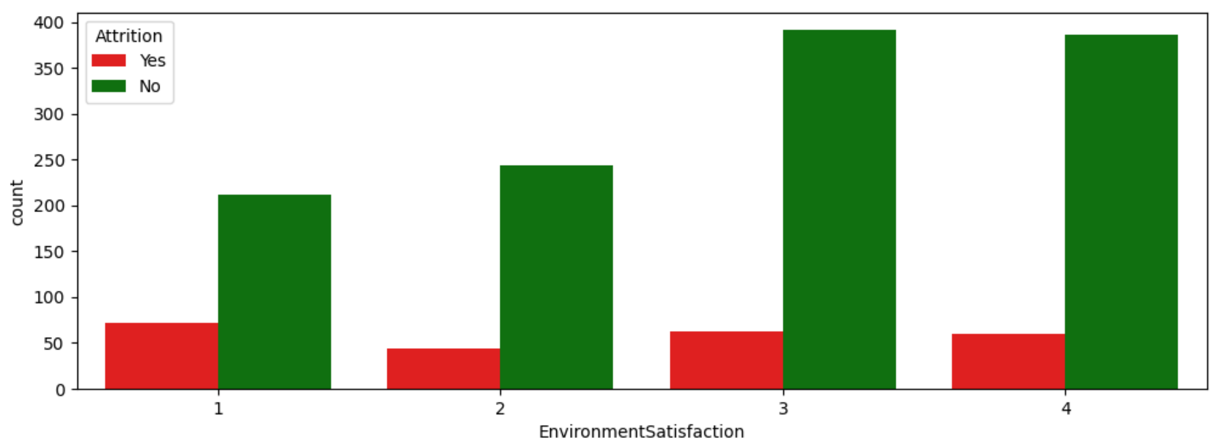

Attrition
Yes
No
count
Travel_Rarely
Travel_Frequently
Non-Travel
BusinessTravel

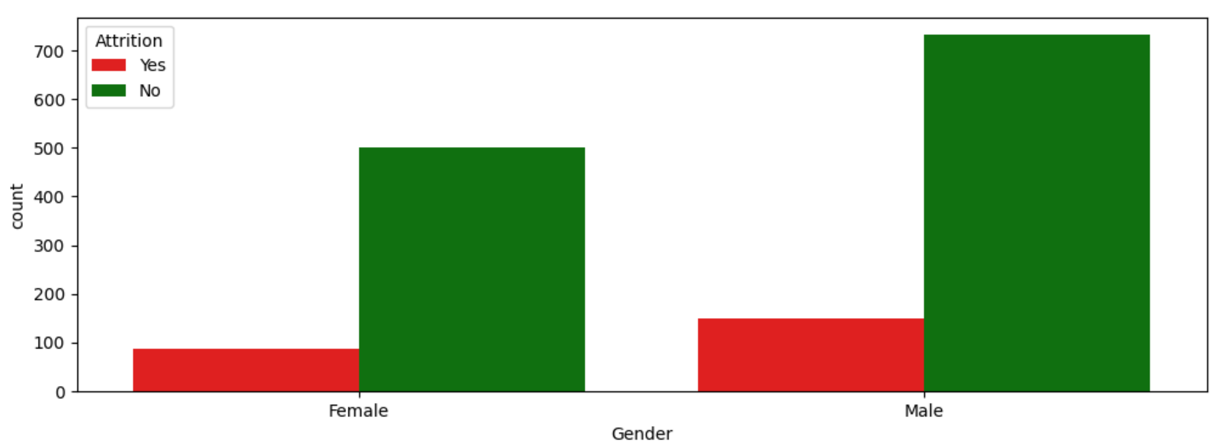

Attrition
Yes
No
count
Sales
Research & Development
Human Resources
Department

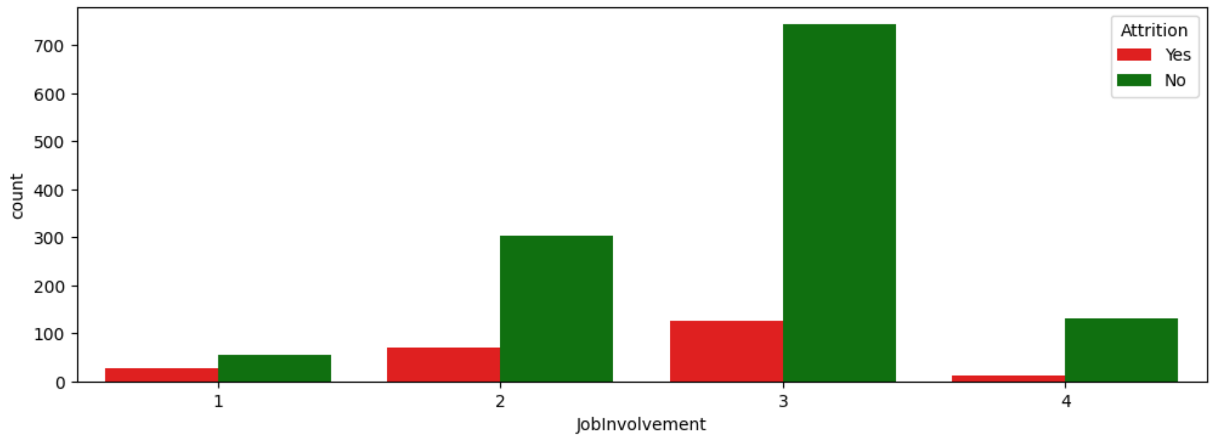

Attrition
Yes
No
count
DistanceFromHome


Attrition
Yes
No
count
0
100
200
300
400
1
2
3
4
5
Education

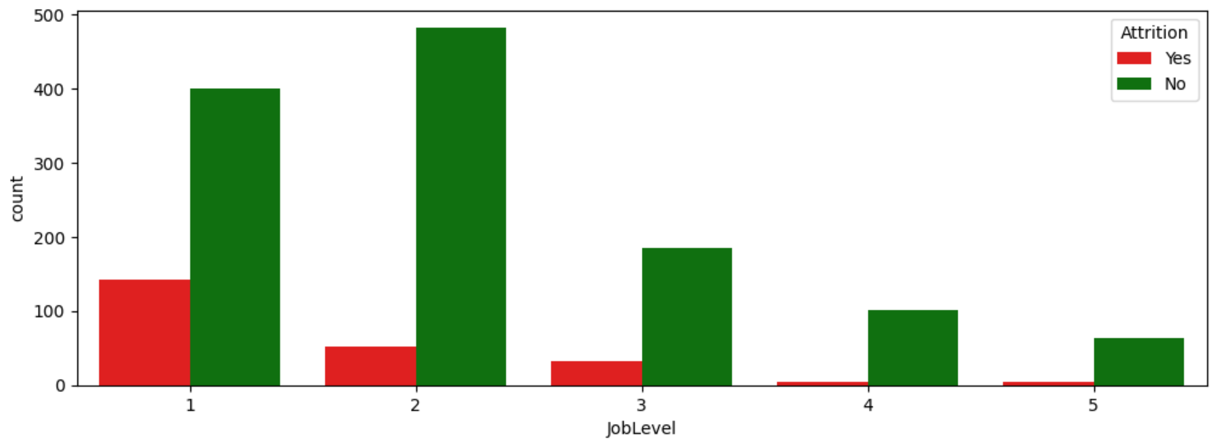

Attrition
Yes
No
count
0
100
200
300
400
500
Life Sciences
Other
Medical
Marketing
Technical Degree
Human Resources
EducationField


Attrition
Yes
No
count
0
50
100
150
200
250
300
350
400
1
2
3
4
EnvironmentSatisfaction

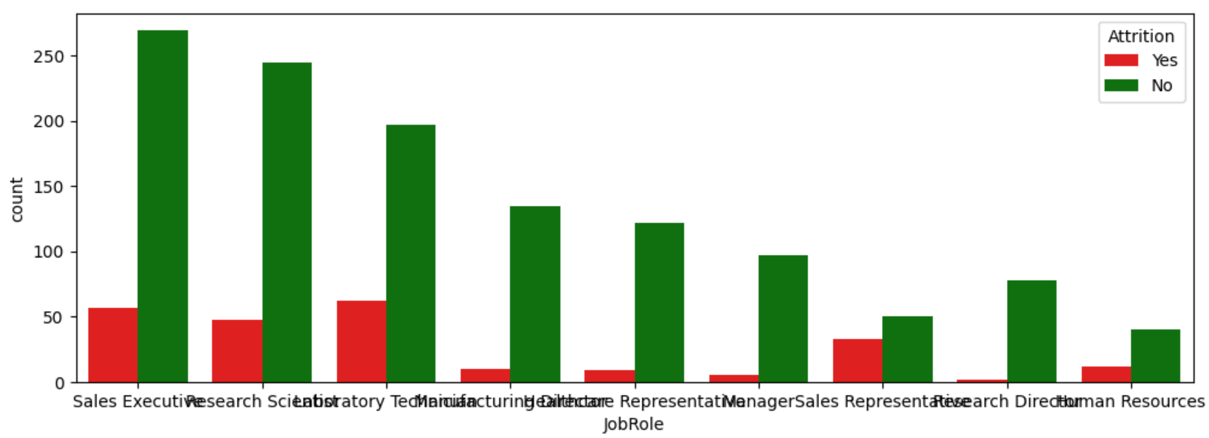

Attrition
Yes
No
count
0
100
200
300
400
500
600
700
Female
Male
Gender



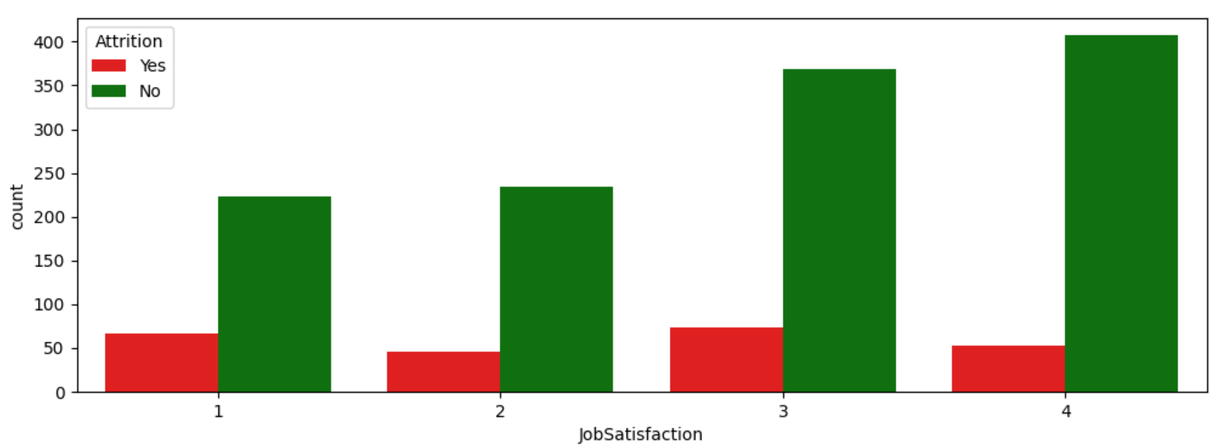
count
JobInvolvement
Attrition
Yes
No



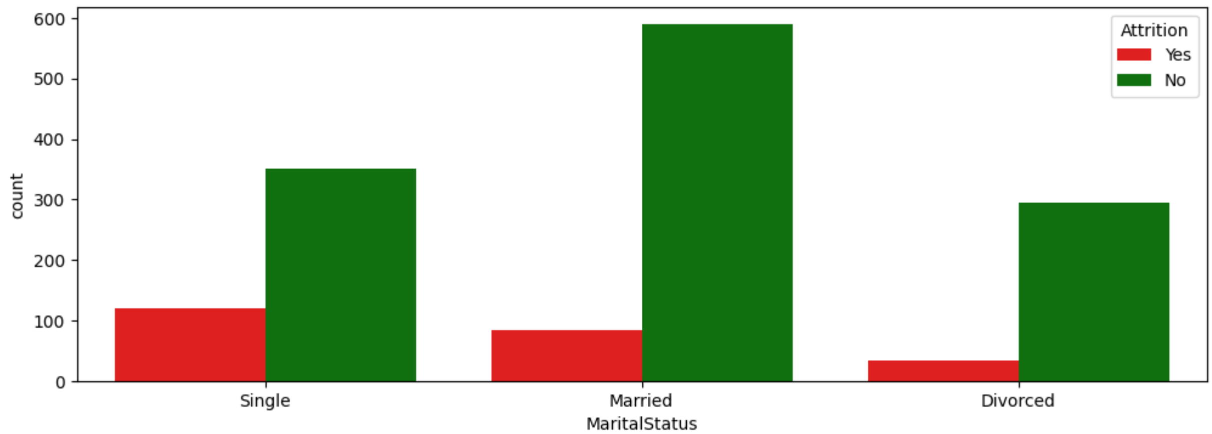
count
JobLevel
Attrition
Yes
No

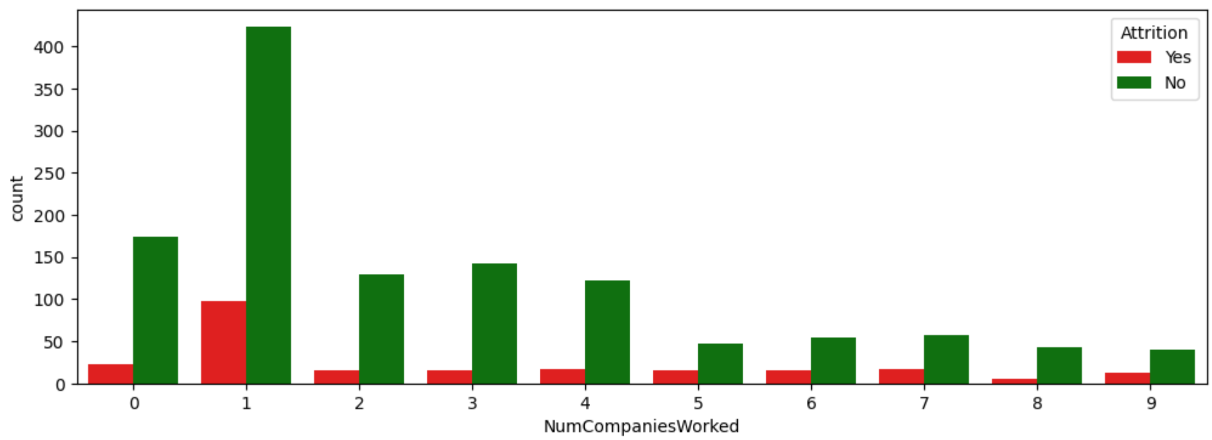


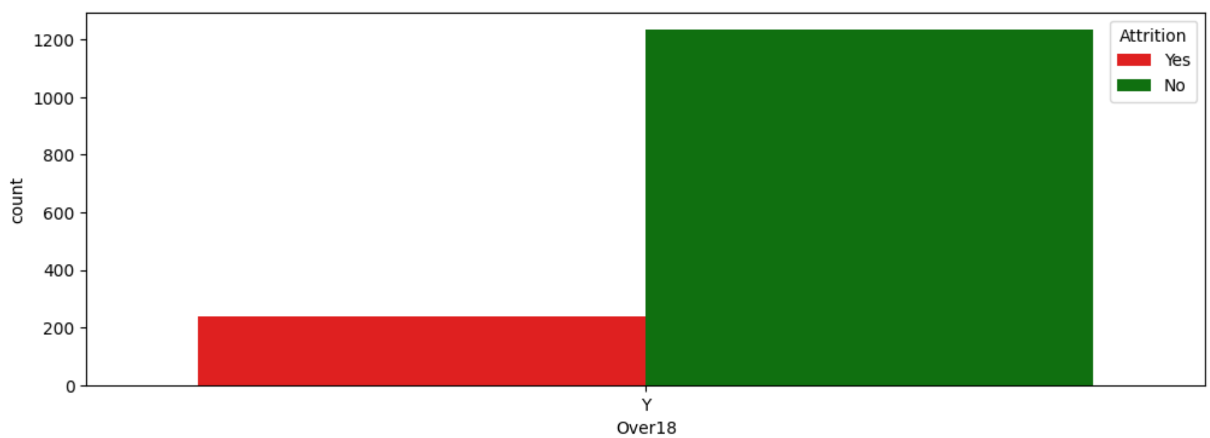
count
JobRole
Attrition
Yes
No
Human Resources

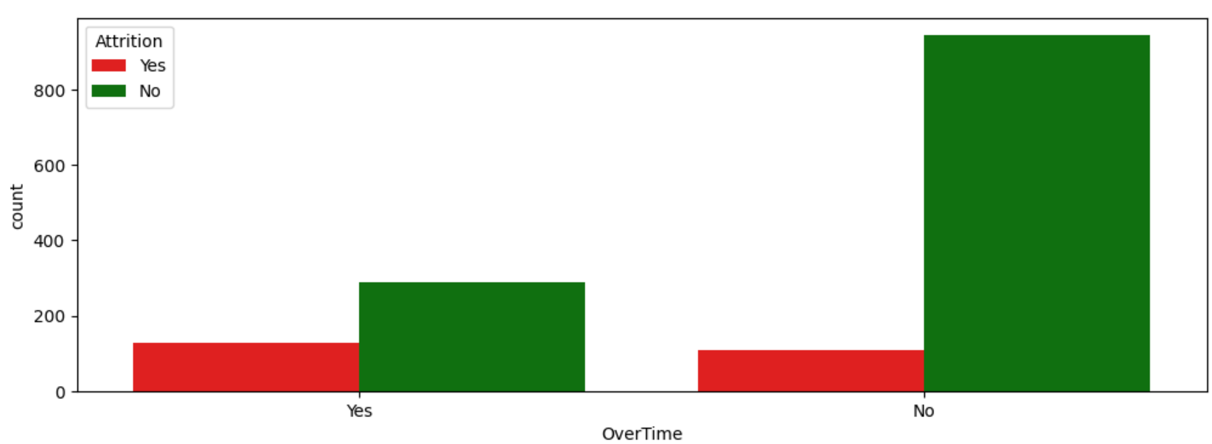


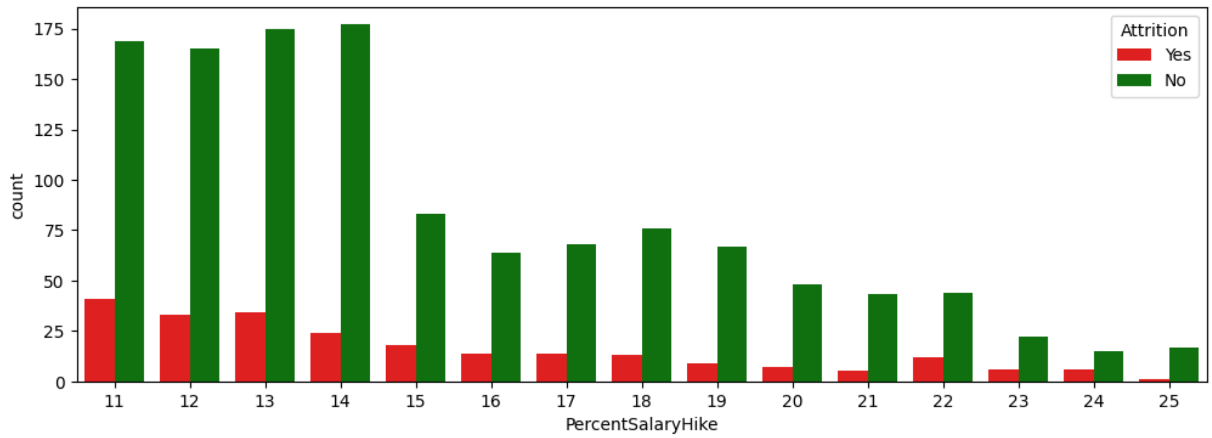
count
JobSatisfaction
Attrition
Yes
No

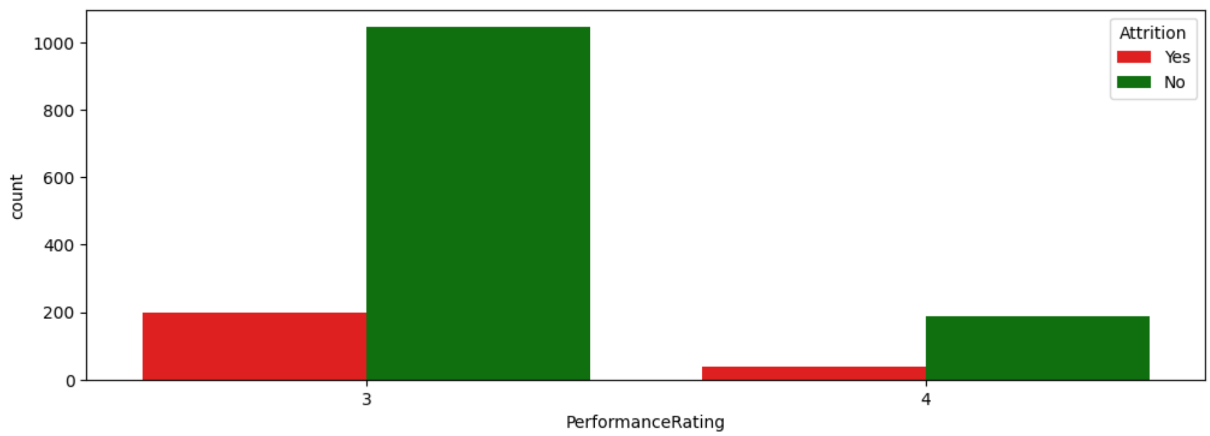

Attrition
Yes
No
count
0
100
200
300
400
500
600
Single
Married
Divorced
MaritalStatus

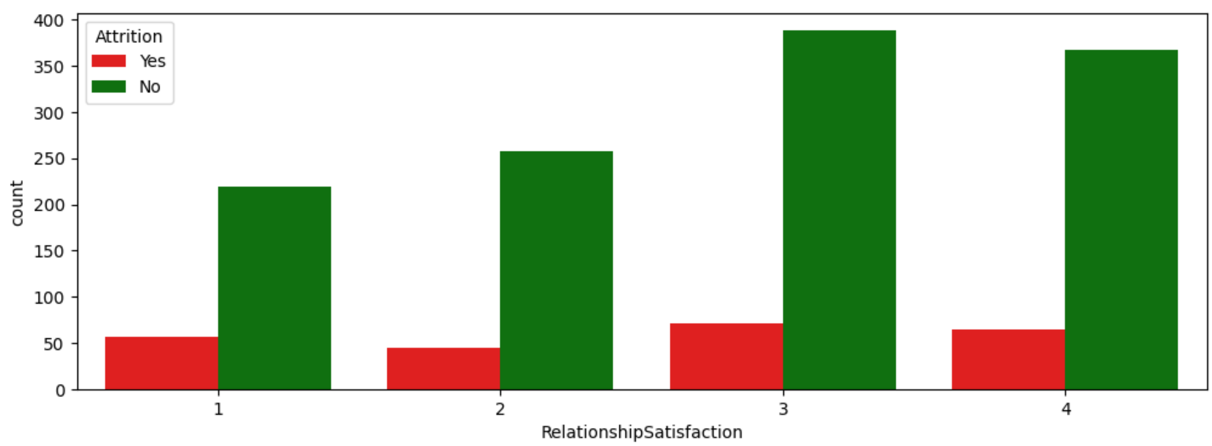

Attrition
Yes
No
count
0
50
100
150
200
250
300
350
400
0
1
2
3
4
5
6
7
8
9
NumCompaniesWorked

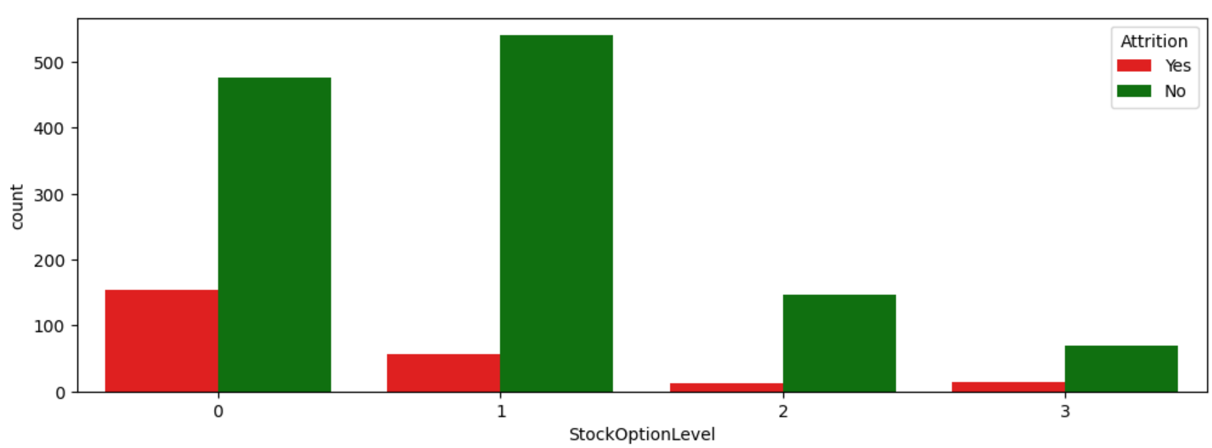

Attrition
Yes
No
count
0
200
400
600
800
1000
1200
Y
Over18

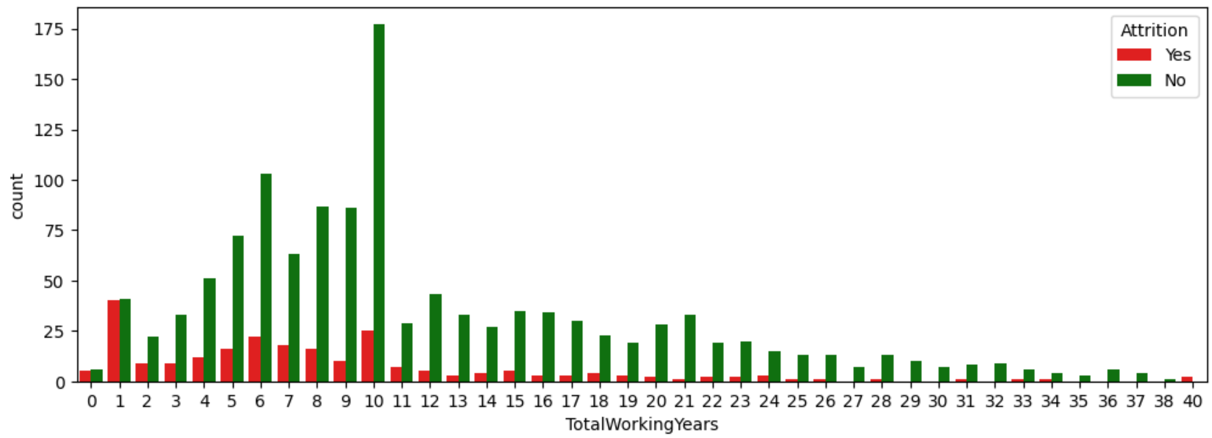

Attrition
Yes
No
count
0
200
400
600
800
Yes
No
OverTime

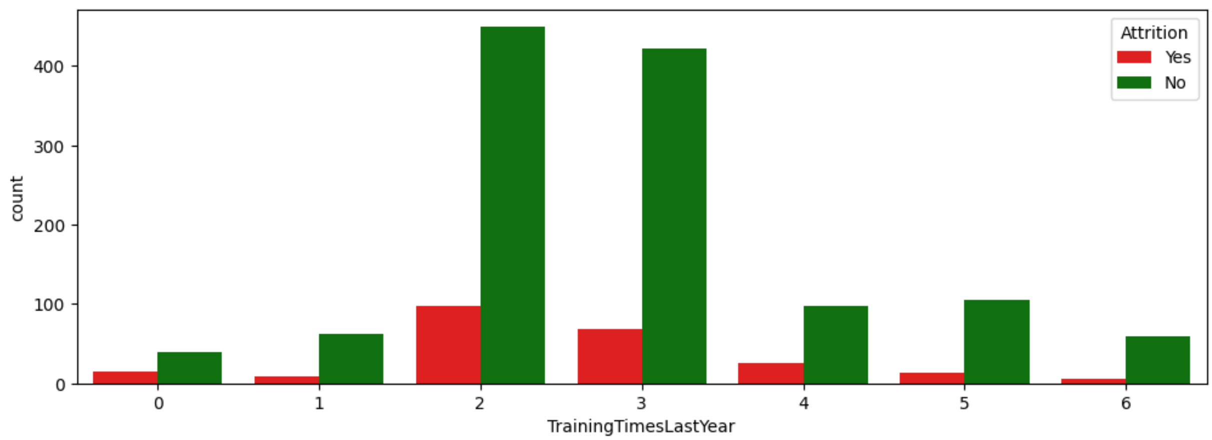

Attrition
Yes
No
count
PercentSalaryHike

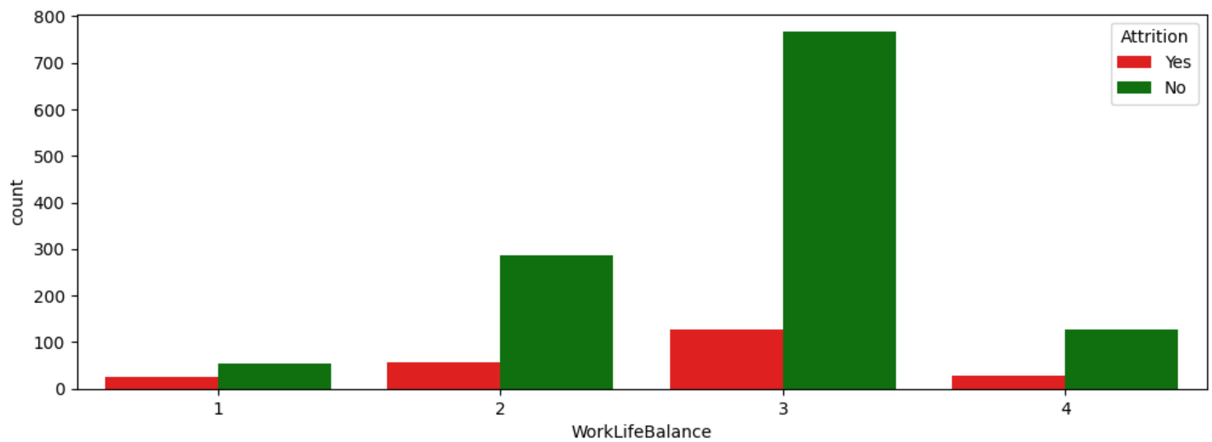

Attrition
Yes
No
count
PerformanceRating

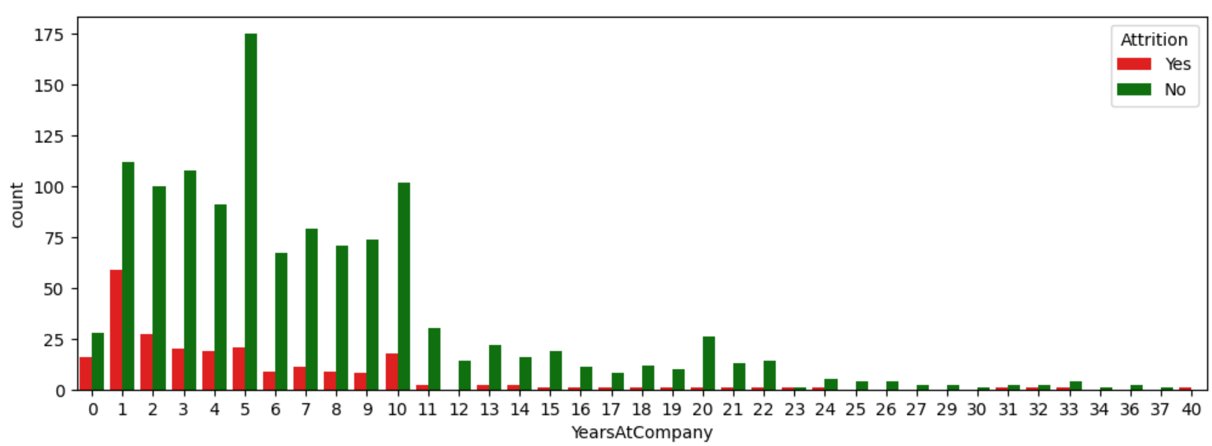

Attrition
Yes
No
count
RelationshipSatisfaction

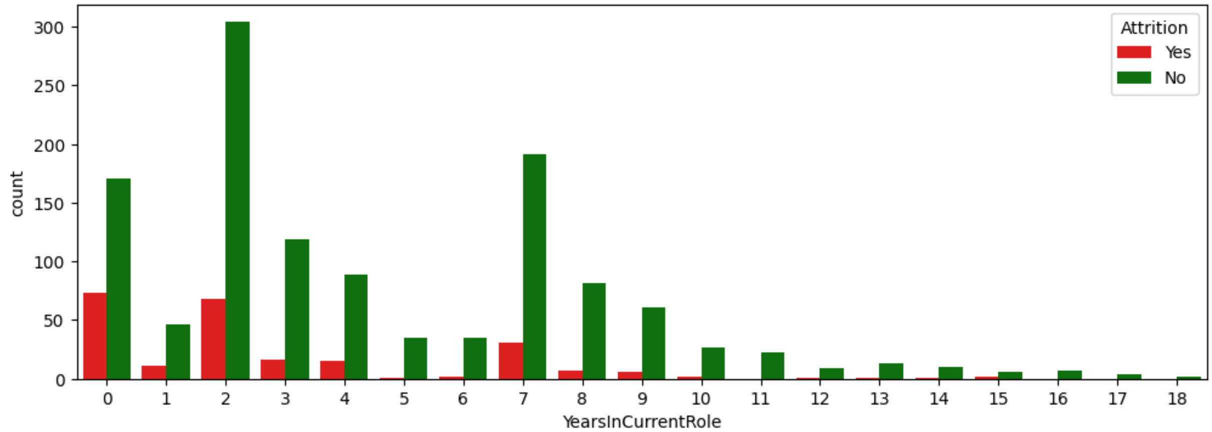

Attrition
Yes
No
count
StockOptionLevel

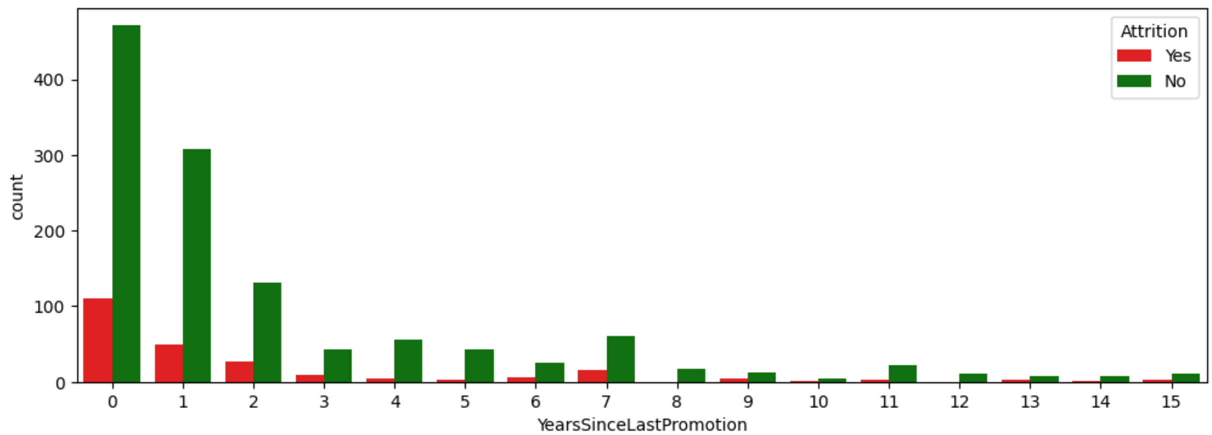

Attrition
Yes
No
count
TotalWorkingYears
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 40
0 25 50 75 100 125 150 175

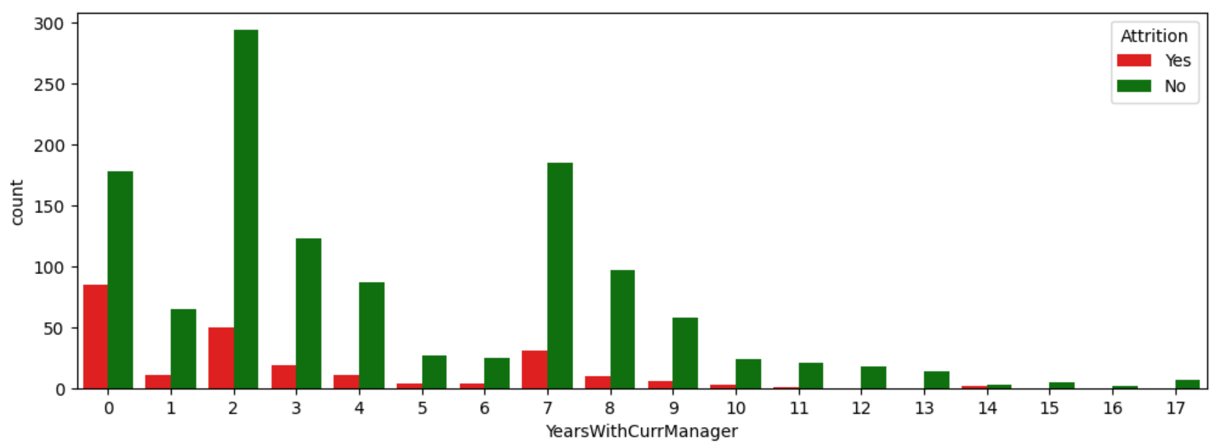

Attrition
Yes
No
count
TrainingTimesLastYear
0 1 2 3 4 5 6
0 100 200 300 400


Attrition
Yes
No
count
WorkLifeBalance
1 2 3 4
0 100 200 300 400 500 600 700 800


Attrition
Yes
No
count
YearsAtCompany
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 29 30 31 32 33 34 36 37 40
0 25 50 75 100 125 150 175







At the first look we can see most on Attrition are the newer employers .

Clean The Data

In [ ]:
```
def wrangle (filepath):
    df = pd.read_csv(filepath)
    df.drop(['EmployeeCount','EmployeeNumber','DailyRate','StandardHours','Over18']
    df['Attrition'] = df['Attrition'].map({'Yes':1, 'No':0})
    df['OverTime'] = df['OverTime'].map({'Yes':1, 'No':0})
    df['BusinessTravel'] = df['BusinessTravel'].map({'Travel_Rarely':1, 'Travel_Fre

    return df
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 30 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   int64 
 2   BusinessTravel            1470 non-null   int64 
 3   Department                1470 non-null   object
 4   DistanceFromHome          1470 non-null   int64 
 5   Education                 1470 non-null   int64 
 6   EducationField            1470 non-null   object
 7   EnvironmentSatisfaction   1470 non-null   int64 
 8   Gender                    1470 non-null   object
 9   HourlyRate                1470 non-null   int64 
 10  JobInvolvement            1470 non-null   int64 
 11  JobLevel                  1470 non-null   int64 
 12  JobRole                   1470 non-null   object
 13  JobSatisfaction           1470 non-null   int64 
 14  MaritalStatus             1470 non-null   object
 15  MonthlyIncome             1470 non-null   int64 
 16  MonthlyRate               1470 non-null   int64 
 17  NumCompaniesWorked        1470 non-null   int64 
 18  OverTime                  1470 non-null   int64 
 19  PercentSalaryHike         1470 non-null   int64 
 20  PerformanceRating         1470 non-null   int64 
 21  RelationshipSatisfaction  1470 non-null   int64 
 22  StockOptionLevel          1470 non-null   int64 
 23  TotalWorkingYears         1470 non-null   int64 
 24  TrainingTimesLastYear     1470 non-null   int64 
 25  WorkLifeBalance           1470 non-null   int64 
 26  YearsAtCompany            1470 non-null   int64 
 27  YearsInCurrentRole        1470 non-null   int64 
 28  YearsSinceLastPromotion   1470 non-null   int64 
 29  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(25), object(5)
memory usage: 344.7+ KB
```

In [ ]:
```
df = wrangle('WA_Fn-UseC_-HR-Employee-Attrition.csv')
df.head()
```

Out[ ]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationFi
0	41	1	1	Sales	1	2	Life Scien
1	49	0	2	Research & Development	8	1	Life Scien
2	37	1	1	Research & Development	2	2	Ot
3	33	0	2	Research & Development	3	4	Life Scien
4	27	0	1	Research & Development	2	1	Med

5 rows × 30 columns

Split the data

In [ ]:
```
X = df.drop('Attrition', axis=1)
y = df['Attrition']
```

In [ ]:
```
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder , OrdinalEncoder
from sklearn.pipeline import make_pipeline
from sklearn.compose import ColumnTransformer
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [ ]:
```
X_train_under, y_train_under = RandomUnderSampler(random_state=42).fit_resample(X_t
```

In [ ]:
```
X_train_over, y_train_over = RandomOverSampler(random_state=42).fit_resample(X_trai
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:
```
categorical_cols = ['Department', 'EducationField', 'Gender', 'JobRole', 'MaritalSt
ordinal_cols = ['Education', 'EnvironmentSatisfaction', 'JobInvolvement', 'JobLevel
                'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel'
numerical_cols = ['Age','BusinessTravel', 'DistanceFromHome', 'HourlyRate', 'Monthl
                  'PercentSalaryHike', 'TotalWorkingYears', 'TrainingTimesLastYear'
                  'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrMa
```

In [ ]:
```
# Create transformers for each type of feature
categorical_transformer = make_pipeline(
    OneHotEncoder(drop='first', sparse=False)  # Use drop='first' to handle multico
```

```
)

ordinal_transformer = make_pipeline(
    OrdinalEncoder()
)

numerical_transformer = make_pipeline(
    StandardScaler()
)

# Create a column transformer to apply the appropriate transformations to each colu
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols),
        ('ord', ordinal_transformer, ordinal_cols)
    ])

# Combine the preprocessing steps and logistic regression into a single pipeline
lr = make_pipeline(
    preprocessor,
    LogisticRegression(random_state=42)
)

# Combine the preprocessing steps and random forest into a single pipeline
rf = make_pipeline(
    preprocessor,
    RandomForestClassifier(random_state=42)
)

# Combine the preprocessing steps and decision tree into a single pipeline
dt = make_pipeline(
    preprocessor,
    DecisionTreeClassifier(random_state=42)
)

# Combine the preprocessing steps and KNN into a single pipeline
knn = make_pipeline(
    preprocessor,
    KNeighborsClassifier()
)
```

In [ ]:

```
# For each pipeline, cross-validate the model on the training data and report the m
pipelines = [lr, rf, dt, knn]
for pipe in pipelines:
    scores = cross_val_score(pipe, X_train_under, y_train_under, cv=5, scoring='acc
    print(f'{pipe.steps[-1][1]}: {scores.mean():.4f} +/- {scores.std():.4f}')
```

```
LogisticRegression(random_state=42): 0.7355 +/- 0.0156
RandomForestClassifier(random_state=42): 0.7013 +/- 0.0505
DecisionTreeClassifier(random_state=42): 0.5979 +/- 0.0342
KNeighborsClassifier(): 0.6429 +/- 0.0329
```

In [ ]:

```
for pipe in pipelines:
    scores = cross_val_score(pipe, X_train, y_train, cv=5, scoring='accuracy')
    print(f'{pipe.steps[-1][1]}: {scores.mean():.4f} +/- {scores.std():.4f}')
```

```
LogisticRegression(random_state=42): 0.8666 +/- 0.0227
RandomForestClassifier(random_state=42): 0.8521 +/- 0.0109
DecisionTreeClassifier(random_state=42): 0.7713 +/- 0.0180
KNeighborsClassifier(): 0.8385 +/- 0.0074
```

In [ ]:
```
for pipe in pipelines:
    scores = cross_val_score(pipe, X_train_over, y_train_over, cv=5, scoring='accur
    print(f'{pipe.steps[-1][1]}: {scores.mean():.4f} +/- {scores.std():.4f}')
```

```
LogisticRegression(random_state=42): 0.7848 +/- 0.0154
RandomForestClassifier(random_state=42): 0.9776 +/- 0.0082
DecisionTreeClassifier(random_state=42): 0.9305 +/- 0.0122
KNeighborsClassifier(): 0.8144 +/- 0.0232
```

for tested data

In [ ]:
```
model = lr.fit(X_train, y_train)
y_pred = model.predict(X_test)
model.score(X_test, y_test)
```

Out[ ]: 0.8967391304347826

In [ ]:
```
model = rf.fit(X_train, y_train)
y_pred = model.predict(X_test)
model.score(X_test, y_test)
```

Out[ ]: 0.8722826086956522

In [ ]:
```
model = dt.fit(X_train, y_train)
y_pred = model.predict(X_test)
model.score(X_test, y_test)
```

Out[ ]: 0.782608695652174

In [ ]:
```
model = knn.fit(X_train, y_train)
y_pred = model.predict(X_test)
model.score(X_test, y_test)
```

Out[ ]: 0.8641304347826086

In [ ]:
```
rf_better = GridSearchCV(
    rf,
    param_grid={
        'randomforestclassifier__n_estimators': [100, 200, 300],
        'randomforestclassifier__max_depth': [None, 5, 10, 15],
        'randomforestclassifier__min_samples_split': [2, 5, 10],
        'randomforestclassifier__min_samples_leaf': [1, 2, 5]
    },
    cv=5,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)
```

In [ ]:
```python
rf_better.fit(X_train_over, y_train_over)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
```

Out[ ]:
```
GridSearchCV
estimator: Pipeline
columntransformer: ColumnTransformer
num                cat              ord
StandardScaler     OneHotEncoder    OrdinalEncoder
RandomForestClassifier
```

In [ ]:
```python
rf_better.best_params_
```

Out[ ]:
```
{'randomforestclassifier__max_depth': None,
 'randomforestclassifier__min_samples_leaf': 1,
 'randomforestclassifier__min_samples_split': 2,
 'randomforestclassifier__n_estimators': 200}
```

In [ ]:
```python
rf_better.score(X_test, y_test)
```

Out[ ]:
```
0.8722826086956522
```

In [ ]:
```python
lr_better = GridSearchCV(
    lr,
    param_grid={
        'logisticregression__C': [0.1, 1, 10, 100],
        'logisticregression__solver': ['lbfgs', 'liblinear']
    },
    cv=5,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)
```

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

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
```

Out[ ]:
```
GridSearchCV
estimator: Pipeline
columntransformer: ColumnTransformer
num                cat              ord
StandardScaler     OneHotEncoder    OrdinalEncoder
LogisticRegression
```

In [ ]:
```python
lr_better.best_params_
```

Out[ ]:
```
{'logisticregression__C': 1, 'logisticregression__solver': 'liblinear'}
```

In [ ]:
```python
lr_better.score(X_test, y_test)
```

Out[ ]:
```
0.8967391304347826
```

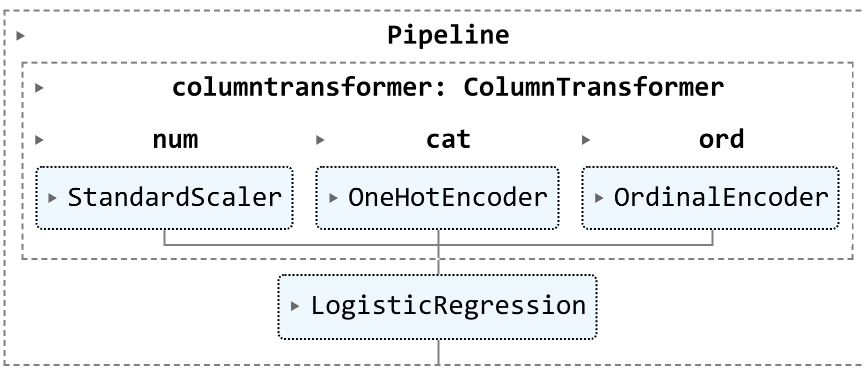
In [ ]:
```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

confusion_matrix(y_test, lr_better.predict(X_test))
```

Out[ ]:
```
array([[308,  12],
       [ 26,  22]], dtype=int64)
```

In [ ]:
```python
final_model = make_pipeline(
    preprocessor,
    LogisticRegression(C=1,solver='liblinear',random_state=42)
)
```

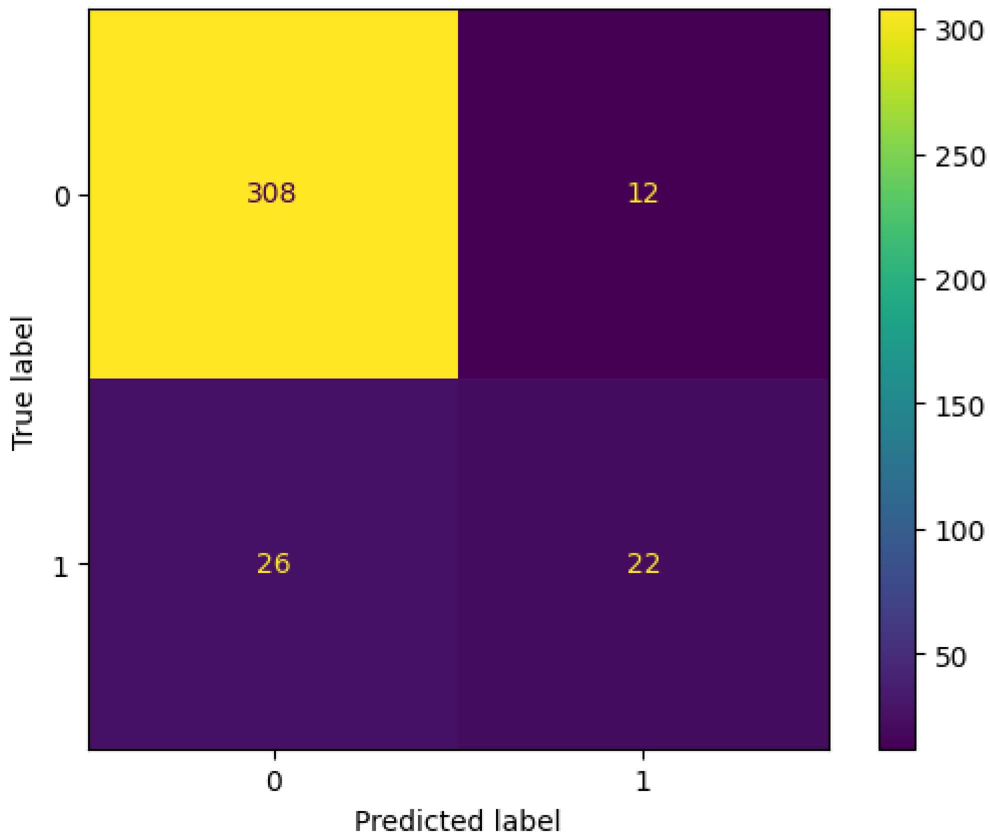
In [ ]:
```python
final_model.fit(X_train, y_train)
```

Out[ ]:



In [ ]:
```python
ConfusionMatrixDisplay.from_estimator(final_model,X_test,y_test)
```

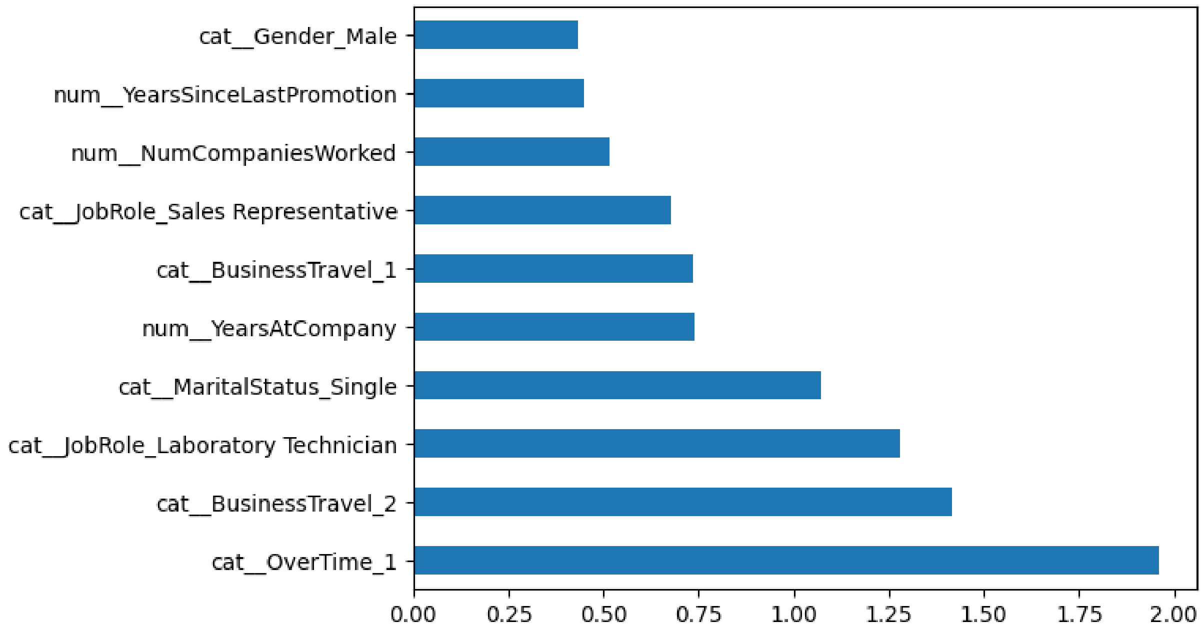
Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x28691bd9bd0>
```



In [ ]:
```
final_model.score(X_test, y_test)
```

Out[ ]: 0.8967391304347826

In [ ]:
```
# the top 10 features that are most important in predicting attrition and plot it
importances = final_model.named_steps['logisticregression'].coef_[0]
features = final_model.named_steps['columntransformer'].get_feature_names_out()
features = np.append(features, numerical_cols)

# Check if the length of importances matches the length of features
if len(importances) < len(features):
    # If there are missing feature importances, set them to zero
    missing_importances = np.zeros(len(features) - len(importances))
    importances = np.concatenate((importances, missing_importances))

feat_importances = pd.Series(importances, index=features)
feat_importances.nlargest(10).plot(kind='barh');
```


cat__Gender_Male
num__YearsSinceLastPromotion
num__NumCompaniesWorked
cat__JobRole_Sales Representative
cat__BusinessTravel_1
num__YearsAtCompany
cat__MaritalStatus_Single
cat__JobRole_Laboratory Technician
cat__BusinessTravel_2
cat__OverTime_1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

In [ ]: