HR Analytics Project- Understanding the Attrition in HR

Problem Statement

Every year a lot of companies hire a number of employees. The companies invest time and money in training those employees, not just this but there are training programs within the compnaies for their existing employees as well.

The objective of the model to increase the effectiveness of their employees and reduce the time and money investing in employees.

HR Analytics:

Human resource analytics (HR analytics) is an area in the field of analtyics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment. HR analtyics does not just deal with gathering data on employee efficiency. Instead, it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.

Attrition in HR

Attrition in human resources refers to the gradual loss of employees overtime. In general, relatively high attrition is problematic for companies. HR professionals often assume a leadership role in designing company compensation programs, work culture, and motivation systems that help the organization retain top employees

How does Attrition affect companies? and how does HR Analytics help in analysing attrition? We will discuss the first question here and for the second question, we will write the code and try to understand the process step by step.

Attrition affecting Companies

A major problem in high employee attrition is its cost to an organization. Job postings, hiring processes, paperwork, and new hire training are some of the common expenses of losing employees and replacing them. Additionally, regular employee turnover prohibits your organization from increasing its collective knowledge base and experience over time. This is especially concerning if your business is customer-facing, as customers often prefer to interact with familiar people. Errors and issues are more likely if you constantly have new workers.

Importing the Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
from matplotlib import pyplot as plt
from scipy.stats import zscore
#data preprocessing
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
#Over Sampling the data using SMOTE
from imblearn.over_sampling import SMOTE
#modelling
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn .ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
from matplotlib import pyplot
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
```

EXPLORATORY DATA ANALYSIS

In [3]:
```
df.describe()
```

Out[3]:

Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	RelationshipSatisfaction	StandardHours
1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	1470.000000	1470.0
2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	2.712245	80.0
1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	1.081209	0.0
1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	1.000000	80.0
2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	2.000000	80.0
3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	3.000000	80.0
4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	4.000000	80.0
5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	4.000000	80.0

Employee's average number of years at company is 7.
Mean is not equal to median stating that the data is not normally distributed. Most normally distributes column is Daily rate where mean is almost equal to median

Out[4]:
```
Age                         int64
Attrition                  object
BusinessTravel             object
DailyRate                   int64
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                     object
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
Over18                     object
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
```

Numeric variables:

- Related to personal information: age, distance_from_home, employee_number
- Related to income: hourly_rate, daily_rate, monthly_rate, monthly_income, percent_salary_hike

Related to duration in company: years_at_company, years_in_current_role, years_since_last_promotion, years_with_curr_manager, total_working_years

num_companies_worked,standard_hourstraining_times_last_year, employee_count

Categorical variables:

- Binary variables: attrition(target variable), gender, over18, over_time
- Nominal variables: department, education_field, job_role, marital_status
- Ordinal variables:

- Ordinal regarding satisfaction and performance:environment_satisfaction,job_satisfaction, relationship_satisfaction,work_life_balance,job_involvement,performance_rating
- Other ordinal: business_travel, education, job_level, stock_option_level

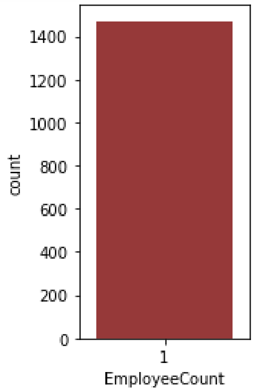
```
__Handling null values

In [5]:  df.isnull().sum()

Out[5]: 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
```

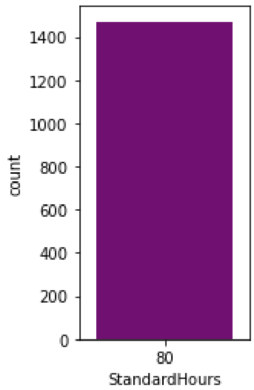
This dataset has no null values

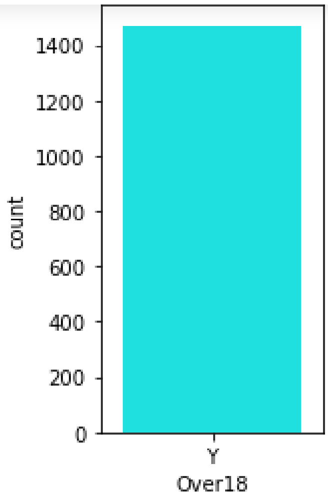
```
In [24]: for col in df:
             print(col)
             print(df[col].value_counts())
             print()

1    1470
Name: EmployeeCount, dtype: int64

EmployeeNumber
2048    1
1368    1
1364    1
1363    1
1362    1
        ..
648     1
647     1
645     1
644     1
2046    1
Name: EmployeeNumber, Length: 1470, dtype: int64
```

Displaying value count of unique value in each feature. To identify column having single unique value

UNI VARIATE ANALYSIS







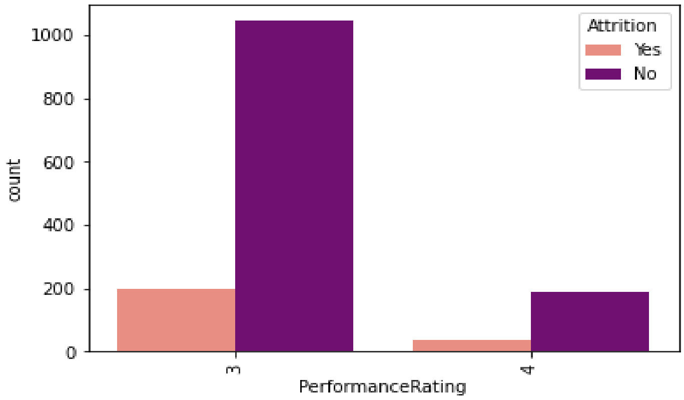
The features standard hours, over18 and employee count has only one value so it won't create any impact on the target feature Attrition.

All the three columns having single value so I'm going to dropping it from the given dataset

BI VARIATE ANALYSIS (Categorical columns vs Target)

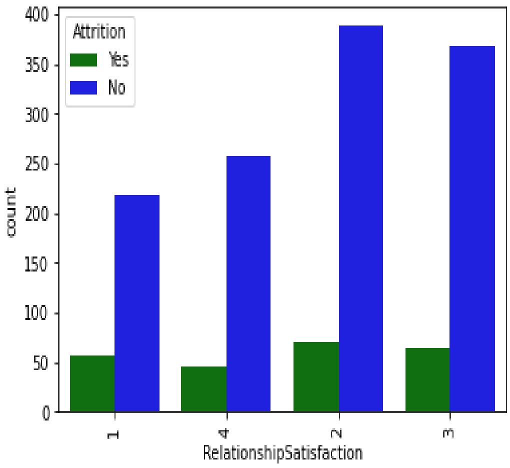
```
l = list(df['PerformanceRating'].unique())
col_l=["salmon","purple"]
chart = sns.countplot(df["PerformanceRating"],palette=col_l,hue=df.Attrition)
chart.set_xticklabels(labels=l, rotation=90)
```

```
[Text(0, 0, '3'), Text(1, 0, '4')]
```

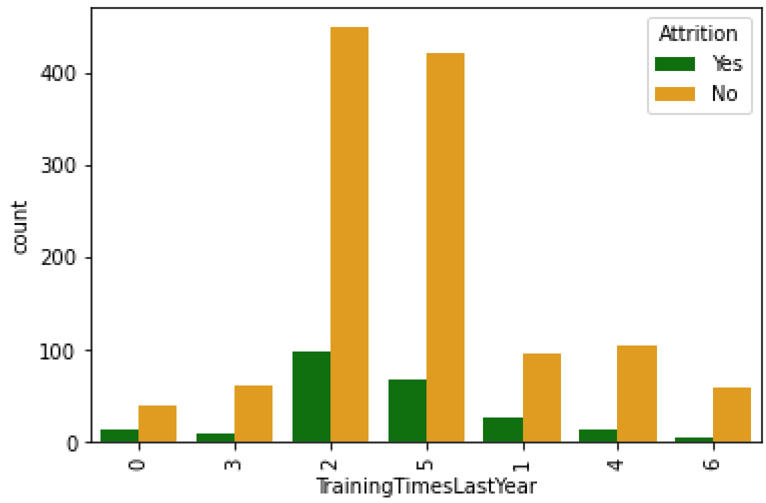



```
the employees who got performance rating as 3 are having less attrition rate.
```

```
[Text(0, 0, '1'), Text(1, 0, '4'), Text(2, 0, '2'), Text(3, 0, '3')]
```

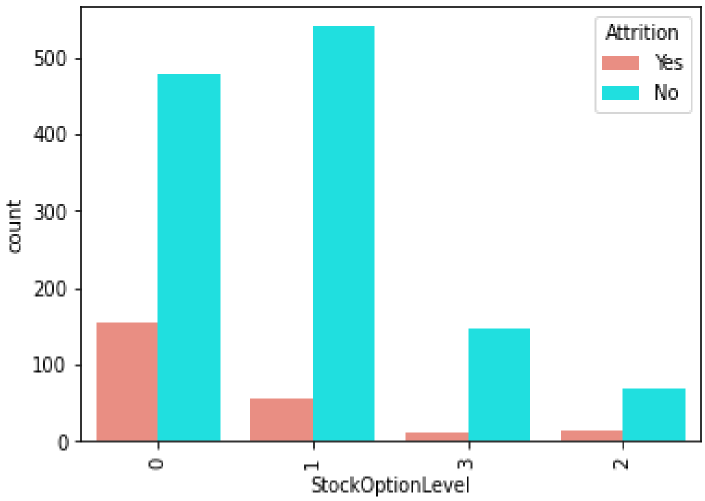


from the above chart its apparent that the employee who having high relationship satisfaction and low relationship satisfaction are having low Attrition rate.so we cannot predict the target coloumn with this value

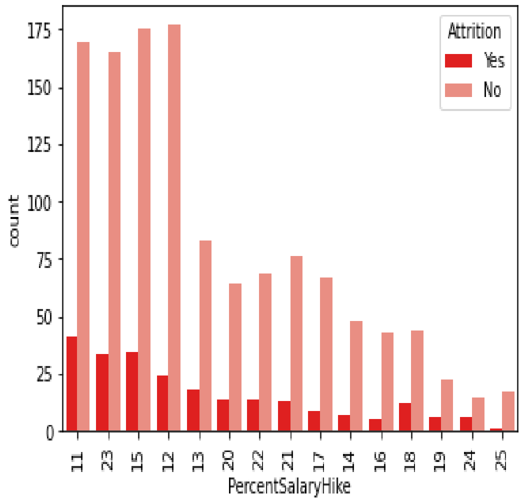


Employee who are trained 2 and 5 times a year are having less attrition rate

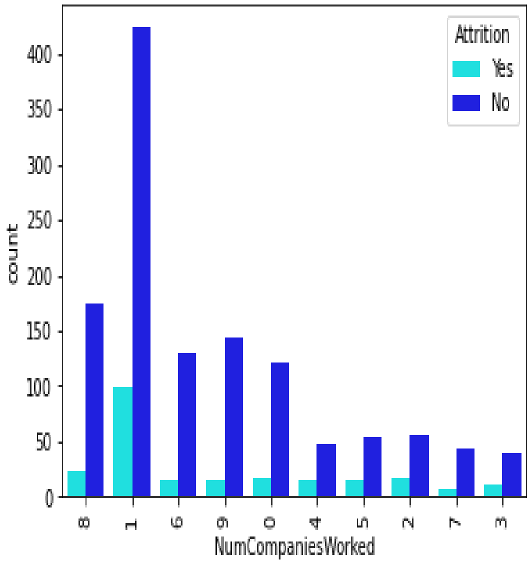
```
]: [Text(0, 0, '0'), Text(1, 0, '1'), Text(2, 0, '3'), Text(3, 0, '2')]
```



the employees who having less stocks are having low attrition rate.



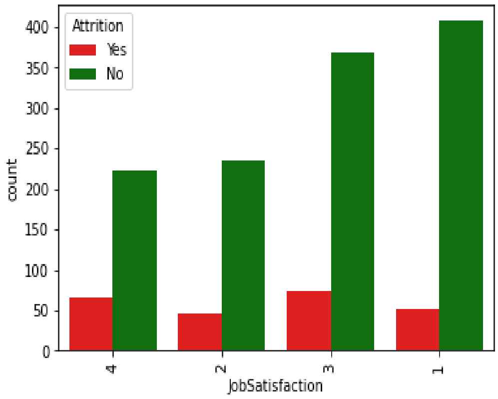
the employees who all are got hike 11 to 15% are having less attrition rate.the employees who got hike between 18 t0 20% having high attrition rate .we can predict the target column using this feature



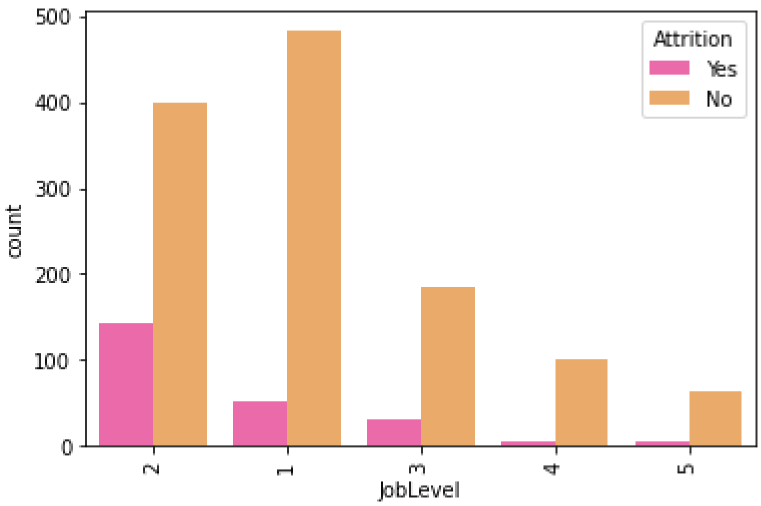
from the above chart its apparent that both the employee who having high satisfaction and low satisfaction are having low Attrition rate.so we cannot predict the target coloumn with this value

from the above chart its apparent that the employees who worked in only one company are having low Attrition rate.

```
[Text(0, 0, '4'), Text(1, 0, '2'), Text(2, 0, '3'), Text(3, 0, '1')]
```



from the above chart its apparent that the employee who having high job satisfaction and low job satisfaction are having low Attrition rate.so we cannot predict the target coloumn with this value

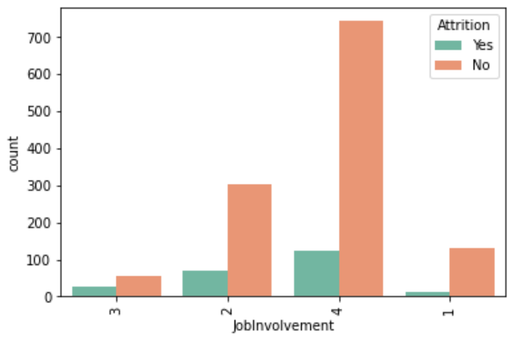


---

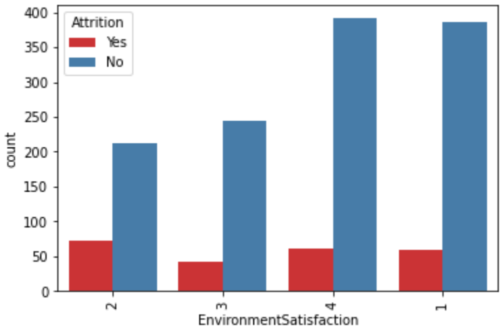
from the above chart its apparent that the entry level employees are having low Attrition rate.

```
33]: l = list(df['JobInvolvement'].unique())
     chart = sns.countplot(df["JobInvolvement"],palette="Set2",hue=df.Attrition)
     chart.set_xticklabels(labels=l, rotation=90)
```

```
33]: [Text(0, 0, '3'), Text(1, 0, '2'), Text(2, 0, '4'), Text(3, 0, '1')]
```

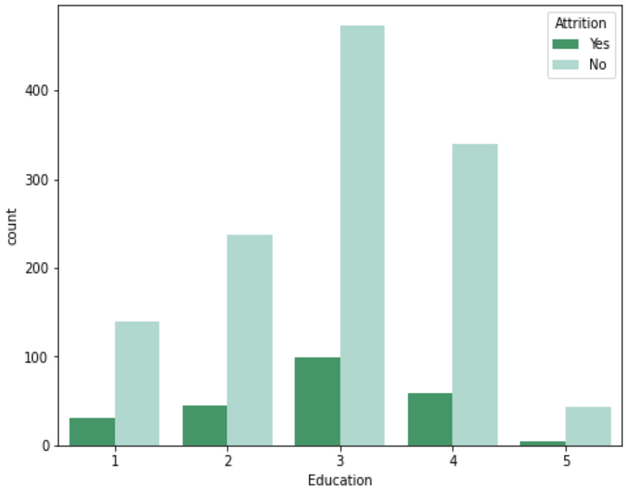


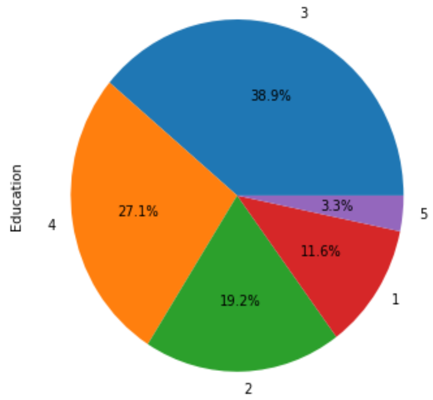
from the above chart its apparent that the employee who all are highly involved in the job having low Attrition rate



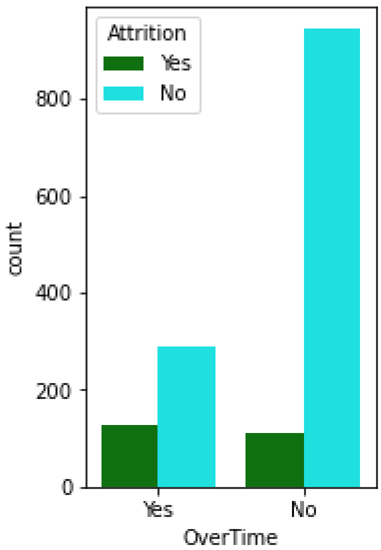
from the above chart its apparent that both the employee who having high satisfaction and low satisfaction are having low Attrition rate.so we cannot predict the target coloumn with this value

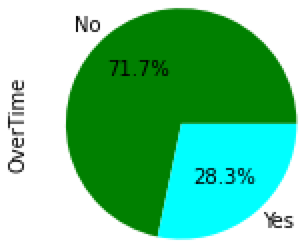
```
<AxesSubplot:ylabel='Education'>
```





The employees of 38.9% belongs to education category 3.compare to other category 3 has less percentange of people moving out of the company



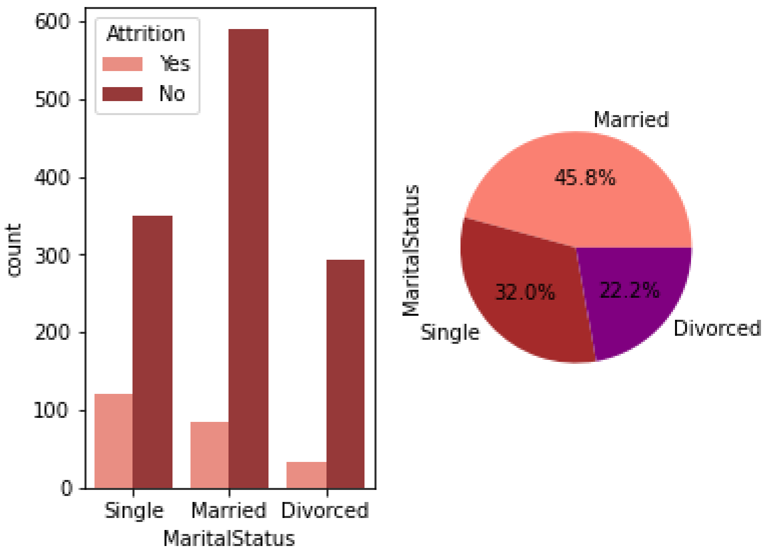


Employees who all are not working overtime has low attrition rate

28.3% employees are willing to work in overtime

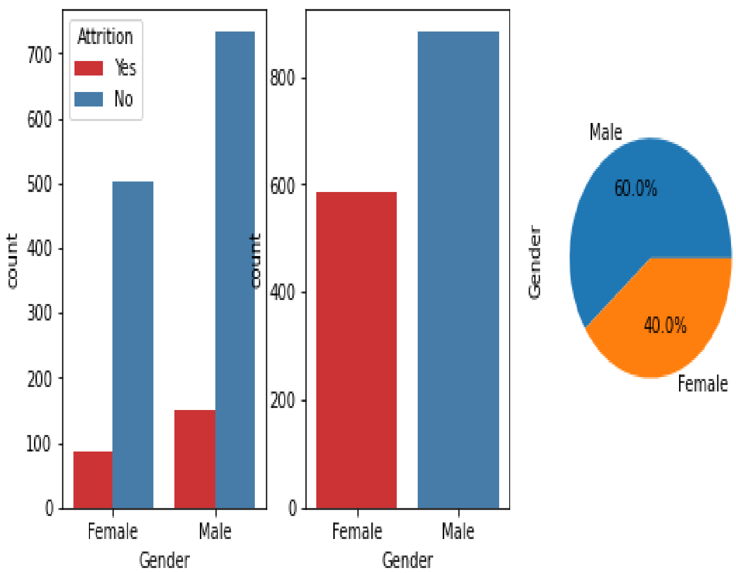
```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

ut[12]: <AxesSubplot:ylabel='MaritalStatus'>

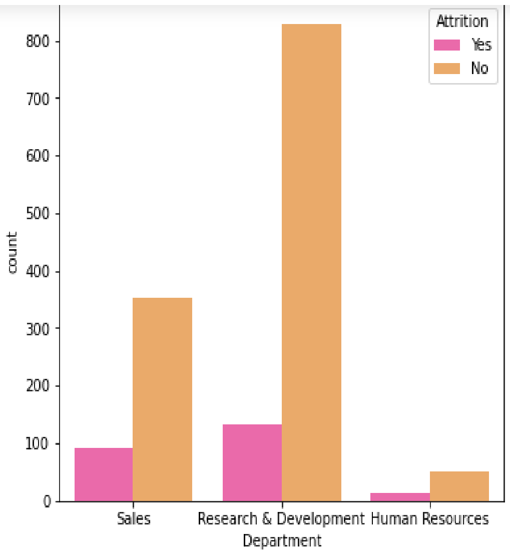


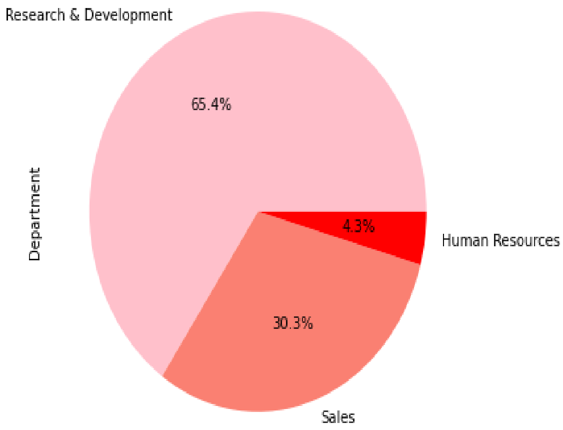
from this above chart its apparent that employees who all are married are having less attrition rate

]: <AxesSubplot:ylabel='Gender'>



comparing the percentage of attrition out of 588 female only 88 people are quitting it means ((((588-88)/588)*100)) 15% female are leaving but in male out of 882 people more than 150 peple are quitting it means(((882-150)/882)*100)18% male are leaving

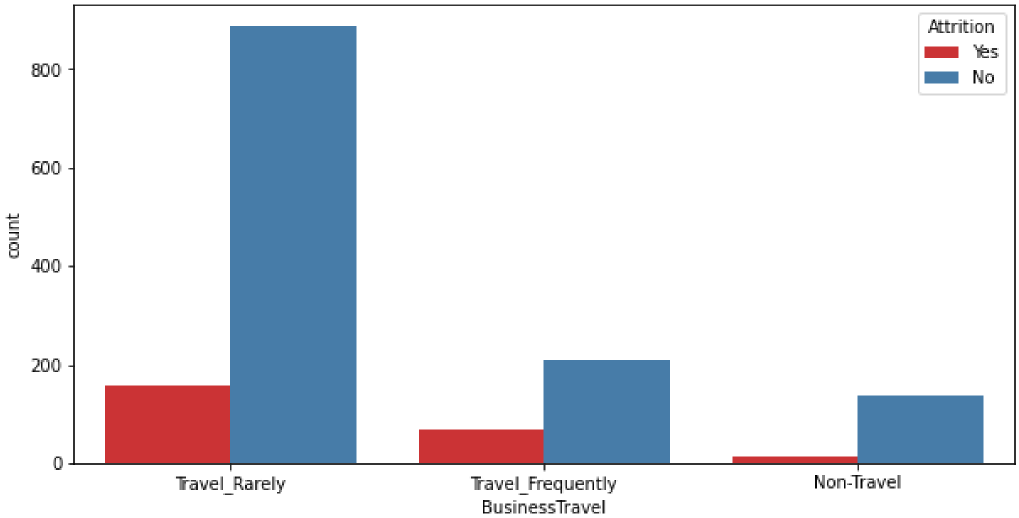




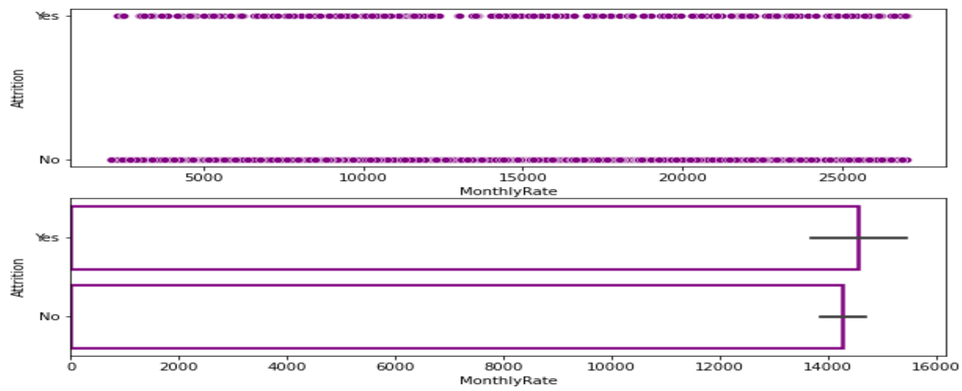
from the above plot its apparent that comparitively the employees who belongs to Research and Developement will like to continue their job.The majority percentage(65.4%) of employee belongs to R&D department

[38]:
```
plt.figure(figsize=(10,5))
sns.countplot(df.BusinessTravel,palette="Set1",hue=df.Attrition)
```

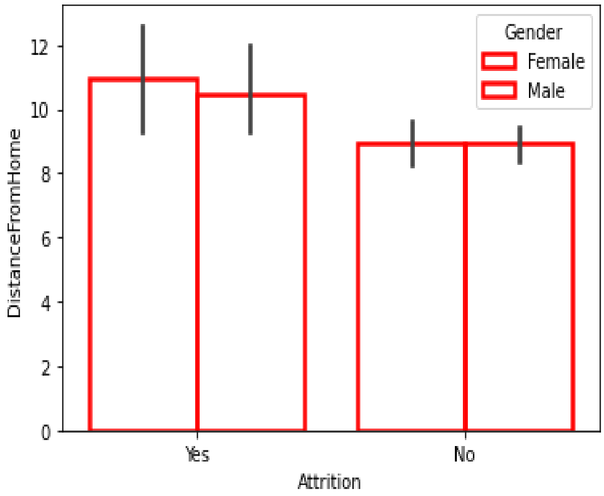
[38]: `<AxesSubplot:xlabel='BusinessTravel', ylabel='count'>`



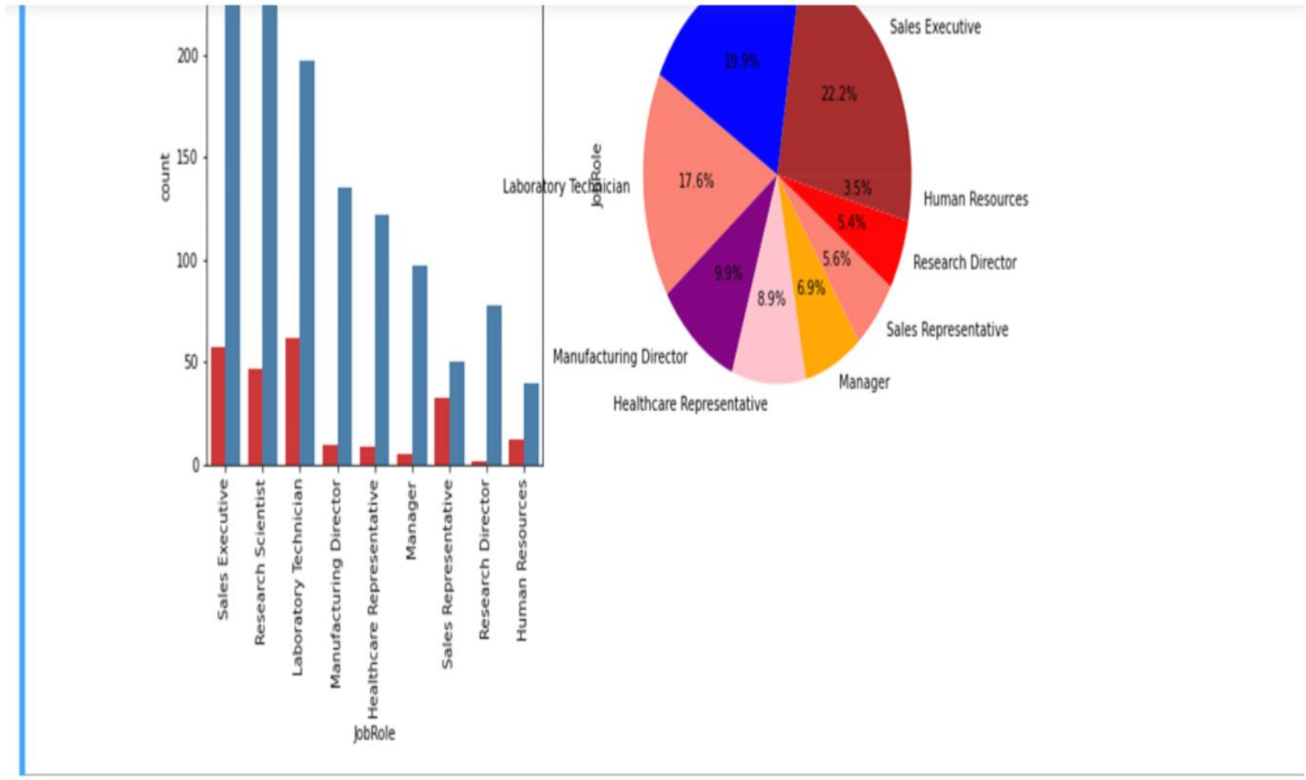
from the above plot its apparent that comparitively the employees who travel rarely will not resign their job



Employees having low monthlyrate are mostly resigning their job



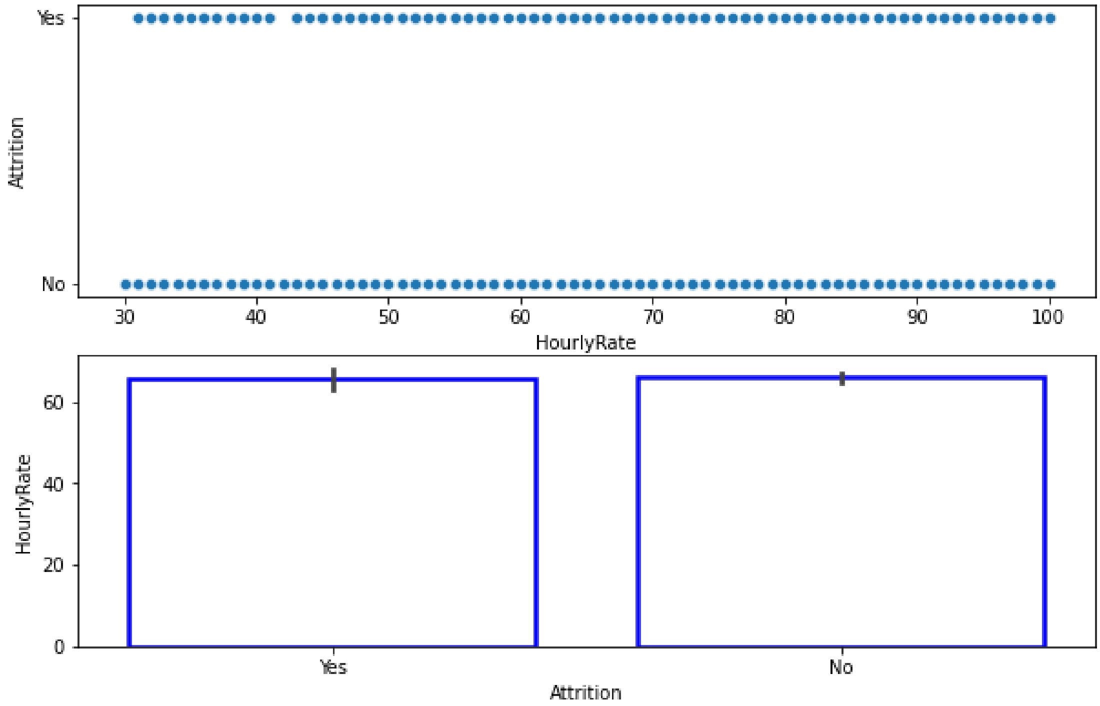
Distance from home is not an important feature to create impact on Attrition feature.Distance from home is not impact on gende



Attrition rate is lower in salesexecutive and research scientist roles

if the employee is in their 5th year with company with same manager are not quitting the job

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



HourlyRate is not an important feature to create impact on Attrition feature

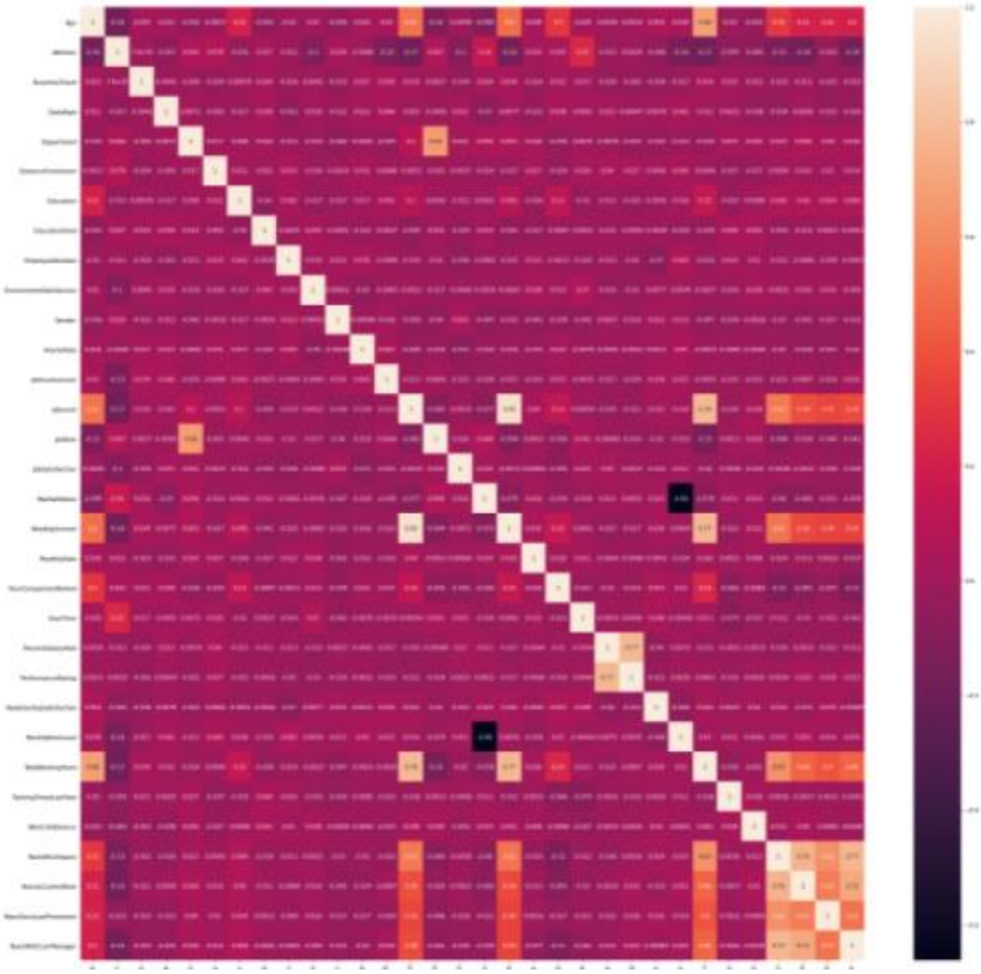
EDA CONCLUSION:

Employees who belongs to below category having less attritionon rate

- travel rarely
- who belongs to R&D department
- who belongs to life science and medical field
- female
- working as sales executive and research scientists
- unmarried
- not working over time
- moderate work life balance
- high job involvement
- working in single company
- performance rating:3
- employees who having 0 stocks
- low monthly income.

we cannot predict using relationship satisfaction,job satisfaction,Environment satisfaction features

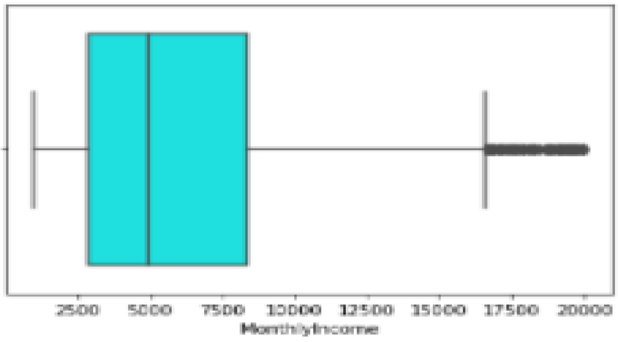
Correlation



Over time feature is highly correlated with attrition

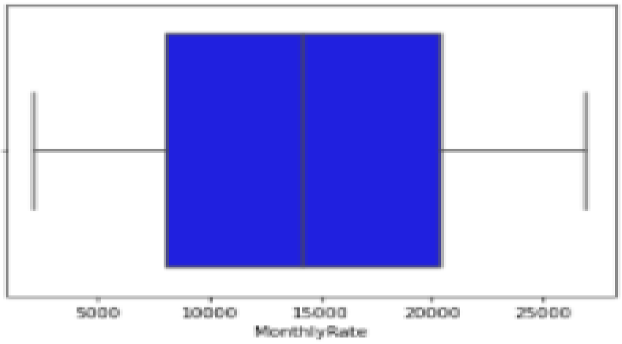
Handling outliers in numerical column:

Out[7]: <AxesSubplot:xlabel='MonthlyIncome'>



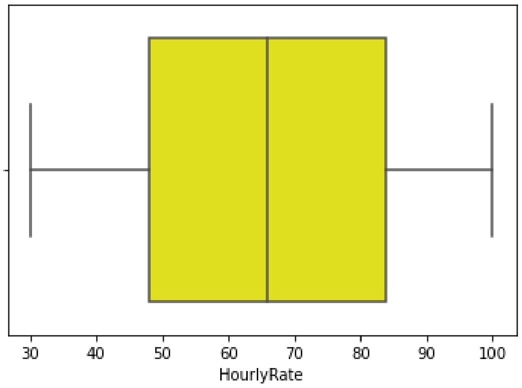
In [8]:
```
sns.boxplot(df['MonthlyRate'],color="blue")
```

Out[8]: <AxesSubplot:xlabel='MonthlyRate'>



In [9]:
```
sns.boxplot(df['DailyRate'],color="green")
```

Out[10]: <AxesSubplot:xlabel='HourlyRate'>



In [11]:
```
z1 = np.abs(stats.zscore(df_new['MonthlyIncome']))
print(z1)
```

[0.10834951 0.29171859 0.93765369 ... 0.07669019 0.23647414 0.44597809]

In [14]:
```
df_new['MonthlyIncome'] = df_new.MonthlyIncome[(z1<3)]
df_new.shape
```

Out[14]: (1470, 32)

_outliers are removed from numerical data monthly income

Data pre-processing:

The features standard hours, over18 and employee count has only single value so it won't create any impact on the target feature Attrition.

Employee Number feature is just an identifier and it's not required for modelling either. So I'm dropping these features

The features standardhours,over18 and employeecount has only one value so it wont create any impact on the target feature Attrition.

In [4]:
```python
cols=['StandardHours','Over18','EmployeeCount']
df_new=df.drop(cols,axis=1)
```

All the three columns having single value so I'm dropping it from the given dataset

In [4]:
```python
df_new.shape
```

Out[4]: (1470, 32)

Encoding all categorical column into numerical column using label encoding technique

In [22]:
```python
data_clean=df_new
col_encod=['Attrition','BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','OverTime']
```

In [23]:
```python
from sklearn import preprocessing
for col in col_encod:
    label = preprocessing.LabelEncoder()
    data_clean[col]= label.fit_transform(df_new[col])
```

In [24]:
```python
data_clean.head(5)
```

Out[24]:

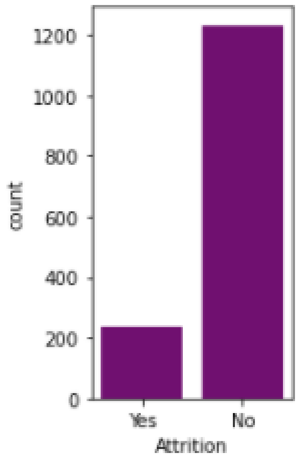
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	...	Perform
0	41	1	2	1102	2	1	2	1	1	2	...	
1	49	0	1	279	1	8	1	1	2	3	...	
2	37	1	2	1373	1	2	2	4	4	4	...	
3	33	0	1	1392	1	3	4	1	5	4	...	
4	27	0	2	591	1	2	1	3	7	1	...	

5 rows × 32 columns

HANDLING CLASS IMBALANCE

Classification problem where the distribution of examples across the known classes is biased or skewed. To avoid this we are using SMOTE technique

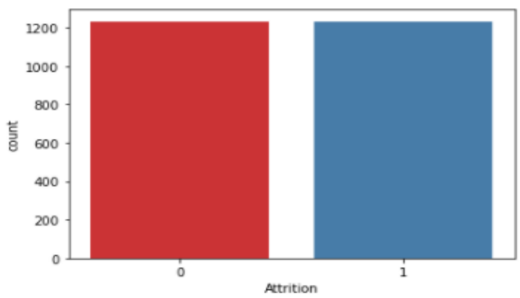
```
In [112]:  plt.figure(figsize=(2,4))
           sns.countplot(df.Attrition,color="purple")

Out[112]: <AxesSubplot:xlabel='Attrition', ylabel='count'>
```



the target coloumn attrition has two values 0 and 1.It has class imbalance

SMOTE synthetic over-sampling works to cause the classifier to build larger decision regions that contain nearby minority class points. This will in turn avoid data loss

```
In [333]:  x1=data.drop('Attrition',axis=1)
           y1=data['Attrition']

In [334]:  x1,y1=over.fit_resample(x1,y1)

In [335]:  sns.countplot(y1,palette="Set1")

Out[335]: <AxesSubplot:xlabel='Attrition', ylabel='count'>
```



Scaling using min max scaler

```
In [336]:  from sklearn.preprocessing import MinMaxScaler
           scaler=MinMaxScaler()
           scaled = scaler.fit_transform(x1)
```

Modelling

It is a binary classification problem so I have modelled using logistic regression and other classification models

___MODELING

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn .ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
from matplotlib import pyplot
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
x=scaled
y=y1
```

```
:rain_test_split(x,y,test_size=0.25,random_state=1)

),RandomForestClassifier(),BaggingClassifier(),KNeighborsClassifier(),GradientBoostingClassifier(),DecisionTreeClassifier()]


```

```
----------------------------------------------------------------------------
KNeighborsClassifier()

Accuracy score: 0.8

"Confusion Matrix:
" [[196 101]
 [ 21 299]]
classification_report
               precision    recall  f1-score   support

           0       0.90      0.66      0.76       297
           1       0.75      0.93      0.83       320

    accuracy                           0.80       617
   macro avg       0.83      0.80      0.80       617
weighted avg       0.82      0.80      0.80       617

Average accuracy_score 0.8022690437601296
```

```
------------------------------------------------------------------------
BaggingClassifier()

Accuracy score: 0.84

"Confusion Matrix:
" [[262  35]
 [ 63 257]]
classification_report
               precision    recall  f1-score   support

           0       0.81      0.88      0.84       297
           1       0.88      0.80      0.84       320

    accuracy                           0.84       617
   macro avg       0.84      0.84      0.84       617
weighted avg       0.84      0.84      0.84       617

Average accuracy_score 0.8411669367909238
------------------------------------------------------------------------
```

```
RandomForestClassifier()

Accuracy score: 0.9

"Confusion Matrix:
" [[276  21]
 [ 43 277]]
classification_report
               precision    recall  f1-score   support

           0       0.87      0.93      0.90       297
           1       0.93      0.87      0.90       320

    accuracy                           0.90       617
   macro avg       0.90      0.90      0.90       617
weighted avg       0.90      0.90      0.90       617

Average accuracy_score 0.8962722852512156
```

```
LogisticRegression()

Accuracy score: 0.8

"Confusion Matrix:
" [[245  52]
 [ 69 251]]
classification_report
              precision    recall  f1-score   support

           0       0.78      0.82      0.80       297
           1       0.83      0.78      0.81       320

    accuracy                           0.80       617
   macro avg       0.80      0.80      0.80       617
weighted avg       0.81      0.80      0.80       617

Average accuracy_score 0.8038897893030794
-----------------------------------------------------------------
```

```
GradientBoostingClassifier()

Accuracy score: 0.88

"Confusion Matrix:
" [[268  29]
 [ 46 274]]
classification_report
              precision    recall  f1-score   support

           0       0.85      0.90      0.88       297
           1       0.90      0.86      0.88       320

    accuracy                           0.88       617
   macro avg       0.88      0.88      0.88       617
weighted avg       0.88      0.88      0.88       617

Average accuracy_score 0.8784440842787682
-----------------------------------------------------------------
```

```
-----------------------------------------------------------------
DecisionTreeClassifier()

Accuracy score: 0.79

"Confusion Matrix:
" [[231  66]
 [ 64 256]]
classification_report
              precision    recall  f1-score   support

           0       0.78      0.78      0.78       297
           1       0.80      0.80      0.80       320

    accuracy                           0.79       617
   macro avg       0.79      0.79      0.79       617
weighted avg       0.79      0.79      0.79       617

Average accuracy_score 0.7893030794165316
```

Random forest classifier has highest accuracy is **0.896272**

Cross Validation:

In order to avoid over fitting, Cross-validation is used to estimate the skill of a machine learning model on unseen data.

```
In [51]: scorel=[]

In [52]: lr=LogisticRegression()
         scores=cross_val_score(lr,x,y,cv=5)
         scorel.append(scores)
         scores

Out[52]: array([0.64574899, 0.85395538, 0.83975659, 0.86206897, 0.84178499])

In [53]: rf=RandomForestClassifier()
         scores=cross_val_score(rf,x,y,cv=5)
         scorel.append(scores)
         scores

Out[53]: array([0.73481781, 0.95537525, 0.93103448, 0.95537525, 0.94523327])

In [54]: bg=BaggingClassifier()
         scores=cross_val_score(bg,x,y,cv=5)
         scorel.append(scores)
         scores

Out[54]: array([0.70445344, 0.9127789 , 0.91075051, 0.90872211, 0.89655172])

In [55]: kn=KNeighborsClassifier()
         scores=cross_val_score(kn,x,y,cv=5)
         scorel.append(scores)
         scores

Out[55]: array([0.76923077, 0.831643  , 0.81541582, 0.81541582, 0.82758621])

In [56]: gb=GradientBoostingClassifier()
         scores=cross_val_score(gb,x,y,cv=5)
         scorel.append(scores)
         scores

Out[56]: array([0.65587045, 0.93711968, 0.90872211, 0.9148073 , 0.9148073 ])
```

Difference of predicted model and crossvalidation score:

- LogisticRegression() difference is0.0378952
- RandomForestClassifier() difference is 0.05058173
- BaggingClassifier() difference is 0.06024702
- KNeighborsClassifier() difference is 0.02531716
- GradientBoostingClassifier() difference is 0.03636322
- DecisionTreeClassifier() difference is 0.05733428

from the observation KNeighborsClassifier model has least difference so I'm selecting KNeighborsClassifier as best model

Hyper Tuning:

__Hyper Tuning

In [226]:
```
from sklearn.model_selection import GridSearchCV,KFold
params = {
    'n_neighbors' : [5,7,9,11,13,15],
               'weights' : ['uniform','distance'],
               'metric' : ['minkowski','euclidean','manhattan'],
                'p':[1,2],'leaf_size':list(range(1,20))

}

gs2 = GridSearchCV(KNeighborsClassifier(), params, verbose = 1, cv=3, n_jobs = -1)
gs2.fit(xtrain, ytrain)
print('Best param:', gs2.best_params_)
```

```
Fitting 3 folds for each of 1368 candidates, totalling 4104 fits
Best param: {'leaf_size': 1, 'metric': 'minkowski', 'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
```

Best parameters: {'leaf_size': 1, 'metric': 'minkowski', 'n_neighbors': 5, 'p': 1, 'weights': 'distance'}

Modelling using best parameter and best model:

Run Code

After Hypertuning

```
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size=.25)
model =KNeighborsClassifier(metric='minkowski', n_neighbors=5,weights='distance',p=1,leaf_size=1)
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

```
0.9027552674230146
```

```
y_pred_1 = model.predict(x_test)
```

```
result = confusion_matrix(y_test, y_pred_1)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred_1)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred_1)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[242  53]
 [  7 315]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.82      0.89       295
           1       0.86      0.98      0.91       322

    accuracy                           0.90       617
   macro avg       0.91      0.90      0.90       617
weighted avg       0.91      0.90      0.90       617

Accuracy: 0.9027552674230146
```

Final model after hyper tuning with accuracy **0.9027552674230146**

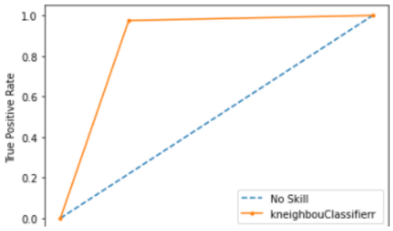
Best model:KNeighbourClassifier Best param: {'leaf_size': 1, 'metric': 'minkowski', 'n_neighbors': 5, 'p': 1, 'weights': 'distance'}

ROC AUC CURVE:

In [68]:

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=1)
m=KNeighborsClassifier(metric='minkowski', n_neighbors=5,weights='distance',p=1,leaf_size=1)
m.fit(xtrain,ytrain)
p=m.predict(xtest)
ns_probs = [0 for _ in range(len(ytest))]
m_probs = p
ns_auc = roc_auc_score(ytest, ns_probs)
m_auc = roc_auc_score(ytest, m_probs)
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('model: ROC AUC=%.3f' % (m_auc))
ns_fpr, ns_tpr,_= roc_curve(ytest, ns_probs)
m_fpr, m_tpr,_=roc_curve(ytest, m_probs)
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(m_fpr, m_tpr, marker='.', label='kneighbouClassifierr ')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()
```

```
No Skill: ROC AUC=0.500
model: ROC AUC=0.878
```



Conclusion:

I have developed a model to predict attrition of an employee with 90.2% accuracy

Saving the model

In [398]:

```
from joblib import dump
dump(model, 'model_hr.joblib')
```

Out[398]: ['model_hr.joblib']

In [399]:

```
from joblib import load
loaded = load('model_hr.joblib')
```