

This project features the implementation of an employee turnover analysis that is built using Python's Scikit-Learn library. In this project, I will use Logistic Regression and Random Forest Machine Learning algorithms. At the end of this project, we would be able to choose the best algorithm for our future projects like Employee Turnover Prediction.

## What is Employee Turnover?

Employee Turnover or Employee Turnover ratio is the measurement of the total number of employees who leave an organization in a particular year. Employee Turnover Prediction means to predict whether an employee is going to leave the organization in the coming period.

A Company uses this predictive analysis to measure how many employees they will need if the potential employees will leave their organization. A company also uses this predictive analysis to make the workplace better for employees by understanding the core reasons for the high turnover ratio.

## Data Preprocessing

Now let's dive into the data to move further with this project on Employee Turnover Prediction

Now let's import the data and move further with the analysis:

```
In [1]:  
import pandas as pd  
from matplotlib import pyplot as plt  
hr = pd.read_csv('E:\Projects\Machine Learning\Employee Turnover Prediction\datasets  
col_names = hr.columns.tolist()  
print("Column names:")  
print(col_names)  
print("\nSample data:")  
hr.head(20)
```

Column names:  
['satisfaction\_level', 'last\_evaluation', 'number\_project', 'average\_montly\_hours',  
'time\_spend\_company', 'Work\_accident', 'left', 'promotion\_last\_5years', 'sales', 'sa  
lary']

Sample data:

```
Out[1]:
```

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company
0	0.38	0.53	2	157	3
1	0.80	0.86	5	262	6
2	0.11	0.88	7	272	4
3	0.72	0.87	5	223	5
4	0.37	0.52	2	159	3
5	0.41	0.50	2	153	3
6	0.10	0.77	6	247	4
7	0.92	0.85	5	259	5
8	0.89	1.00	5	224	5
9	0.42	0.53	2	142	3
10	0.45	0.54	2	135	3
11	0.11	0.81	6	305	4
12	0.84	0.92	4	234	5
13	0.41	0.55	2	148	3
14	0.36	0.56	2	137	3
15	0.38	0.54	2	143	3
16	0.45	0.47	2	160	3
17	0.78	0.99	4	255	6
18	0.45	0.51	2	160	3
19	0.76	0.89	5	262	5

Rename column name from "sales" to "department":

In [2]:

```
hr=hr.rename(columns = {'sales':'department'})
hr
```

Out[2]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	
3	0.72	0.87	5	223	
4	0.37	0.52	2	159	
...	...	...	...	...	...
14994	0.40	0.57	2	151	
14995	0.37	0.48	2	160	
14996	0.37	0.53	2	143	
14997	0.11	0.96	6	280	

satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_compar
14998	0.37	0.52	2	158

14999 rows × 10 columns



In [3]: `hr.shape`

Out[3]: (14999, 10)

The “left” column is the outcome variable recording one and 0. 1 for employees who left the company and 0 for those who didn’t.

The department column of the dataset has many categories, and we need to reduce the categories for better modelling. Let’s see all the categories of the department column:

In [4]: `hr['department'].unique()`

Out[4]: array(['sales', 'accounting', 'hr', 'technical', 'support', 'management', 'IT', 'product\_mng', 'marketing', 'RandD'], dtype=object)

Let’s add all the “technical”, “support” and “IT” columns into one column to make our analysis easier.

In [5]: `import numpy as np  
hr['department']=np.where(hr['department'] =='support', 'technical', hr['department'])  
hr['department']=np.where(hr['department'] =='IT', 'technical', hr['department'])`

## Creating Variables for Categorical Variables

As there are two categorical variables (department, salary) in the dataset and they need to be converted to dummy variables before they can be used for modelling.

In [6]: `cat_vars=['department', 'salary']  
for var in cat_vars:  
 cat_list='var'+'_'+var  
 cat_list = pd.get_dummies(hr[var], prefix=var)  
 hr1=hr.join(cat_list)  
 hr=hr1`

Now the actual variables need to be removed after the dummy variable have been created.

Column names after creating dummy variables for categorical variables:

In [7]: `hr.drop(hr.columns[[8, 9]], axis=1, inplace=True)  
hr.columns.values`

Out[7]: array(['satisfaction\_level', 'last\_evaluation', 'number\_project', 'average\_montly\_hours', 'time\_spend\_company', 'Work\_accident', 'left', 'promotion\_last\_5years', 'department\_RandD', 'department\_accounting', 'department\_hr', 'department\_management', 'department\_marketing', 'department\_product\_mng',

```
'department_sales', 'department_technical', 'salary_high',
'salary_low', 'salary_medium'], dtype=object)
```

The outcome variable is "left", and all the other variables are predictors.

```
In [8]: hr_vars=hr.columns.values.tolist()
y=['left']
X=[i for i in hr_vars if i not in y]
```

## Feature Selection for Employee Turnover Prediction

Let's use the feature selection method to decide which variables are the best option that can predict employee turnover with great accuracy. There are a total of 18 columns in X, and now let's see how we can select about 10 from them:

```
In [9]: from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
rfe = RFE(model, 10)
rfe = rfe.fit(hr[X], hr[y])
print(rfe.support_)
print(rfe.ranking_)
```

```
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass n_features_to_select=10 as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error
    warnings.warn(f"Pass {args_msg} as keyword args. From version "
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max\_iter) or scale the data as shown in:  
<https://scikit-learn.org/stable/modules/preprocessing.html>  
Please also refer to the documentation for alternative solver options:  
[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

```
n_iter_i = _check_optimize_result(
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
```

```
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
[ True  True False False  True  True  True False  True  True False
 False False False  True  True False]
[1 1 3 9 1 1 1 1 5 1 1 6 8 7 4 1 1 2]
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
```

You can see that our feature selection chose the 10 variables for us, which are marked True in the support\_array and marked with a choice "1" in the ranking\_array. Now let's have a look at these columns:

```
In [10]: cols=['satisfaction_level', 'last_evaluation', 'time_spend_company', 'Work_accident',
      'department_RandD', 'department_hr', 'department_management', 'salary_high',
      X=hr[cols]
      y=hr['left']]
```

## Logistic Regression Model to Predict Employee Turnover

```
In [11]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
Out[11]: LogisticRegression()
```

Let's check the accuracy of our logistic regression model.

```
In [12]: from sklearn.metrics import accuracy_score
print('Logistic regression accuracy: {:.3f}'.format(accuracy_score(y_test, logreg.predict(X_test))))
```

```
Logistic regression accuracy: 0.771
```

## Random Forest Classification Model

```
In [13]: from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
Out[13]: RandomForestClassifier()
```

Now let's check the accuracy of our Random Forest Classification Model:

```
In [14]: RandomForestClassifier(bootstrap=True, class_weight=None,
```

```
criterion='gini', max_depth=None, max_features='auto',
max_leaf_nodes=None, min_impurity_split=1e-07,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
oob_score=False, random_state=None, verbose=0, warm_start=False)
```

Out[14]: RandomForestClassifier(min\_impurity\_split=1e-07, n\_estimators=10, n\_jobs=1)

In [15]: `print('Random Forest Accuracy: {:.3f}'.format(accuracy_score(y_test, rf.predict(X_te`)

Random Forest Accuracy: 0.978

## Confusion Matrix for our Machine Learning Models

Now I will construct a confusion matrix to visualize predictions made by our classifier and evaluate the accuracy of our machine learning classification.

### Random Forest

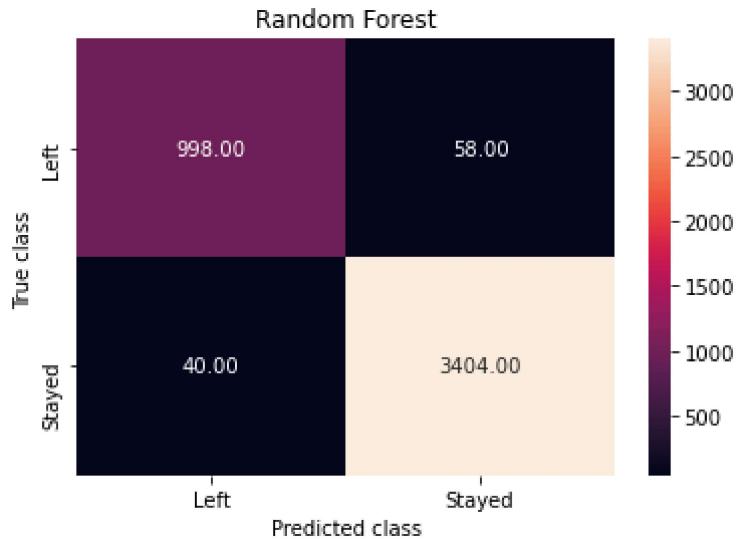
In [16]: `from sklearn.metrics import classification_report
print(classification_report(y_test, rf.predict(X_test)))`

	precision	recall	f1-score	support
0	0.99	0.98	0.99	3462
1	0.95	0.96	0.95	1038
accuracy			0.98	4500
macro avg	0.97	0.97	0.97	4500
weighted avg	0.98	0.98	0.98	4500

In [17]: `y_pred = rf.predict(X_test)
from sklearn.metrics import confusion_matrix
import seaborn as sns
forest_cm = metrics.confusion_matrix(y_pred, y_test, [1,0])
sns.heatmap(forest_cm, annot=True, fmt='.2f', xticklabels = ["Left", "Stayed"] , ytic
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.title('Random Forest')`

```
C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass labels=[1, 0] as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error
    warnings.warn(f"Pass {args_msg} as keyword args. From version "
```

Out[17]: Text(0.5, 1.0, 'Random Forest')



## Logistic Regression

In [18]:

```
print(classification_report(y_test, logreg.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.81	0.92	0.86	3462
1	0.51	0.26	0.35	1038
accuracy			0.77	4500
macro avg	0.66	0.59	0.60	4500
weighted avg	0.74	0.77	0.74	4500

In [19]:

```
logreg_y_pred = logreg.predict(X_test)
logreg_cm = metrics.confusion_matrix(logreg_y_pred, y_test, [1,0])
sns.heatmap(logreg_cm, annot=True, fmt='2f', xticklabels = ["Left", "Stayed"], yticklabels = ["Stayed", "Left"])
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.title('Logistic Regression')
```

C:\Users\Kushal Gupta\anaconda3\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass labels=[1, 0] as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error  
 warnings.warn(f"Pass {args\_msg} as keyword args. From version "

Out[19]: Text(0.5, 1.0, 'Logistic Regression')

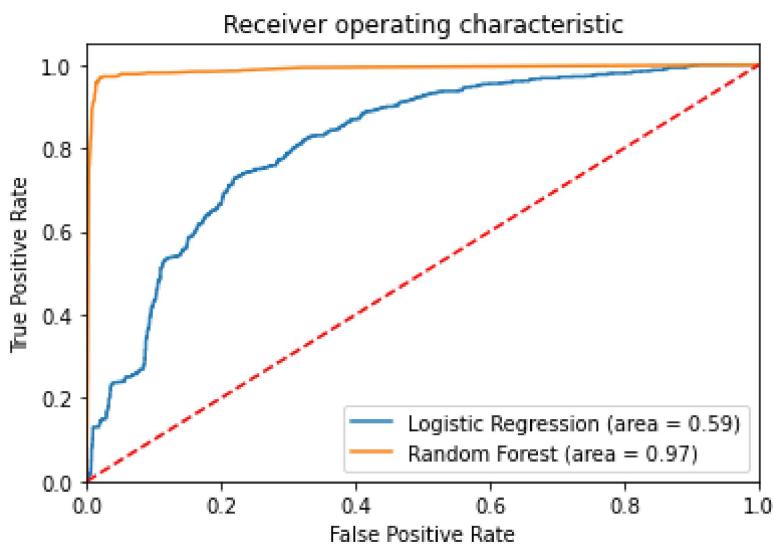


# Employee Turnover Prediction Curve

In [20]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
rf_roc_auc = roc_auc_score(y_test, rf.predict(X_test))
rf_fpr, rf_tpr, rf_thresholds = roc_curve(y_test, rf.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(rf_fpr, rf_tpr, label='Random Forest (area = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```



The receiver operating characteristic (ROC) curve is a standard tool used with binary classifiers. The red dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

So, as we can see that the Random Forest Model has proven to be more useful in the prediction of employee turnover, now let's have a look at the feature importance of our random forest classification model.

In [21]:

```
feature_labels = np.array(['satisfaction_level', 'last_evaluation', 'time_spend_comp',
                           'department_RandD', 'department_hr', 'department_management', 'salary_high',
                           'importance = rf.feature_importances_'
                           feature_indexes_by_importance = importance.argsort()
                           for index in feature_indexes_by_importance:
                               print('{:}-{:2f}%'.format(feature_labels[index], (importance[index] *100.0)))
```

```
promotion_last_5years-0.25%
department_management-0.26%
department_hr-0.30%
department_RandD-0.33%
salary_high-0.71%
salary_low-1.11%
Work_accident-1.48%
last_evaluation-18.41%
time_spend_company-26.50%
satisfaction_level-50.65%
```

**According to our Random Forest classification model, the above aspects show the most important features which will influence whether an employee will leave the company, in ascending order.**

In [ ]: