Importing libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing dataset

```
df = pd.read_csv("Employee.csv")
df.head()
```

	Age	PerNew	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYear	WorkLifeBal
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	80	0	8	0	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	80	1	10	3	3
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	80	0	7	3	3
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	80	0	8	3	3
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	80	1	6	3	3

5 rows × 35 columns

```
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
dtype: int64, object: 9, int64: 26
memory usage: 402.0 KB
```

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	TotalWorkingYears	Tra
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	1470.000000	1470.0	1470.000000	1470.000000	1470.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	2.712245	80.0	0.793878	11.279592	2.799
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	1.081209	0.0	0.852077	7.780782	1.289
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	1.000000	80.0	0.000000	0.000000	0.000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	2.000000	80.0	0.000000	6.000000	2.000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	3.000000	80.0	1.000000	10.000000	3.000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	4.000000	80.0	1.000000	15.000000	3.000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	4.000000	80.0	3.000000	40.000000	6.000

8 rows × 26 columns

```
df.Attrition.value_counts()
```

```
Attrition
No    1233
Yes    237
Name: count, dtype: int64
```

```
df.Department.value_counts()
```

```
Department
Research & Development    961
Sales                     446
Human Resources            63
Name: count, dtype: int64
```

```
df.EducationField.value_counts()
```

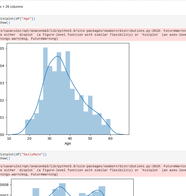
```
EducationField
Life Sciences    606
Medical          464
Marketing        159
Technical Degree 132
Other             82
Human Resources   27
Name: count, dtype: int64
```

Data visualization

```
sns.heatmap(df.isnull(),cmap=False)
plt.show()
df.info()
```

```
<class>
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
dtype: int64, object: 9, int64: 26
memory usage: 402.0 KB
```

```
/Users/saurav/anjali/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

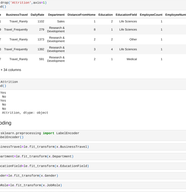



```
sns.distplot(df["DailyRate"])
plt.show()
```

```
/Users/saurav/anjali/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

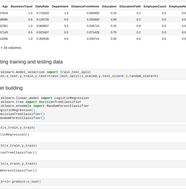
```
sns.boxplot(df.Age)
plt.show()
```

```
/Users/saurav/anjali/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```




```
sns.boxplot(df.DailyRate)
plt.show()
```

```
/Users/saurav/anjali/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Splitting dependent and independent variables

```
df.drop('Attrition',axis=1)
df.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYe
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	2	...	1	80	0	8	0
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	3	...	4	80	1	10	3
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	4	...	2	80	0	7	3
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	4	...	3	80	0	8	3
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	1	...	4	80	1	6	3

5 rows × 34 columns

```
df.Attrition
```

```
0    Yes
1     No
2    Yes
3     No
4     No
Name: Attrition, dtype: object
```

Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df = BusinessTravelle.fit_transform(df.BusinessTravel)
df = DepartmentLe.fit_transform(df.Department)
df = EducationFieldLe.fit_transform(df.EducationField)
df = GenderLe.fit_transform(df.Gender)
df = JobRoleLe.fit_transform(df.JobRole)
df = MaritalStatusLe.fit_transform(df.MaritalStatus)
df = OverTimeLe.fit_transform(df.OverTime)
df = Over18Le.fit_transform(df.Over18)
df.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYe
0	41	2	1102	2	1	2	1	1	1	2	...	1	80	0	8	0
1	49	1	279	1	8	1	1	1	2	3	...	4	80	1	10	3
2	37	2	1373	1	2	2	4	1	4	4	...	2	80	0	7	3
3	33	1	1392	1	3	4	1	1	5	4	...	3	80	0	8	3
4	27	2	591	1	2	1	3	1	7	1	...	4	80	1	6	3

5 rows × 34 columns

```
df.Attrition
```

```
0    Yes
1     No
2    Yes
3     No
4     No
Name: Attrition, dtype: object
```

Feature scaling

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
x_scaled.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYe
0	0.547619	1.0	0.715820	1.0	0.000000	0.25	0.2	0.0	0.000000	0.333333	...	0.333333	0.0	0.000000	0.190476	0.000000
1	0.738095	0.5	0.126782	0.5	0.250000	0.00	0.2	0.0	0.000484	0.666667	...	1.000000	0.0	0.333333	0.250	0.666667
2	0.452381	1.0	0.909807	0.5	0.035714	0.25	0.8	0.0	0.001451	0.666667	...	0.333333	0.0	0.000000	0.170	0.666667
3	0.357143	0.5	0.923407	0.5	0.071429	0.75	0.2	0.0	0.001937	0.666667	...	0.666667	0.0	0.000000	0.200	0.666667
4	0.214286	1.0	0.350036	0.5	0.035714	0.00	0.6	0.0	0.002903	0.000000	...	1.000000	0.0	0.333333	0.100	0.666667

5 rows × 33 columns

Splitting training and testing data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_state=0)
```

Model building

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
lr=LogisticRegression()
dtc=DecisionTreeClassifier()
rfc=RandomForestClassifier()
```

```
lr.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
rfc.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

```
pred_lr=lr.predict(x_test)
```

```
pred_dtc=dtc.predict(x_test)
```

```
pred_rfc=rfc.predict(x_test)
```

```
pred_lr=lr.predict(x_test)
new_df=pd.DataFrame({'Actual':y_test, 'Predicted':pred_lr})
new_df
```

```
# Predicting using Decision tree
new_df=pd.DataFrame({'Actual':y_test, 'Predicted':pred_dtc})
new_df
```

	Actual	Predicted
342	No	No
1065	No	No
1095	No	No
1004	No	No
798	No	No
1332	No	No
...	...	...
1189	No	No
1289	No	No
1162	No	No
1035	No	No
1299	No	No

294 rows × 2 columns

```
# Predicting using Random Forest
new_df=pd.DataFrame({'Actual':y_test, 'Predicted':pred_rfc})
new_df
```

	Actual	Predicted
342	No	No
1065	No	No
1095	No	No
1004	No	No
798	No	No
1332	No	No
...	...	...
1189	No	No
1289	No	No
1162	No	No
1035	No	No
1299	No	No

294 rows × 2 columns

Evaluation metrics

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
accuracy_lr=accuracy_score(y_test,pred_lr)
accuracy_dtc=accuracy_score(y_test,pred_dtc)
accuracy_rfc=accuracy_score(y_test,pred_rfc)
```

```
# Accuracy Score
print("Logistic regression",accuracy_lr)
print("Decision tree",accuracy_dtc)
print("Random Forest",accuracy_rfc)
```

```
Logistic regression 0.8843537414965986
Decision tree 0.7755102040816326
Random Forest 0.8843537414965986
```

```
confusion_tree=confusion_matrix(y_test,pred_lr)
confusion_dtc=confusion_matrix(y_test,pred_dtc)
confusion_rfc=confusion_matrix(y_test,pred_rfc)
```

```
# Confusion matrix
print("Logistic regression",confusion_lr)
print("\n")
print("Decision tree",confusion_dtc)
print("\n")
print("Random Forest",confusion_rfc)
```

```
Logistic regression [[257  3]
 [ 31  3]]

Decision tree [[227  33]
 [ 32  2]]

Random Forest [[257  3]
 [ 31  3]]
```

```
print(classification_report(y_test,pred_lr))
```

```
              precision    recall  f1-score   support

          No       0.89       0.99       0.93       260
         Yes       0.50       0.09       0.15        34

    accuracy                            0.88       294
   macro avg       0.69       0.54       0.54       294
weighted avg       0.85       0.88       0.84       294
```

```
print(classification_report(y_test,pred_dtc))
```

```
              precision    recall  f1-score   support

          No       0.88       0.87       0.87       260
         Yes       0.05       0.06       0.05        34

    accuracy                            0.77       294
   macro avg       0.46       0.46       0.46       294
weighted avg       0.79       0.78       0.78       294
```

```
print(classification_report(y_test,pred_rfc))
```

```
              precision    recall  f1-score   support

          No       0.89       0.99       0.93       260
         Yes       0.50       0.09       0.15        34

    accuracy                            0.88       294
   macro avg       0.69       0.54       0.54       294
weighted avg       0.85       0.88       0.84       294
```