TASK 3: EMPLOYEE ATTRITION PREDICTION USING ML

In [1]:
```
#Import Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:
```
#Store the data into the df variable
df = pd.read_csv('employee.csv')
```

In [3]:
```
df.head(7) #Print the first 7 rows
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educatio
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	I
5	32	No	Travel_Frequently	1005	Research & Development	2	2	Life S
6	59	No	Travel_Rarely	1324	Research & Development	3	3	I

7 rows × 35 columns

In [4]:
```
#Get the number of rows and number of columns in the data
df.shape
```

Out[4]:
```
(1470, 35)
```

In [5]:
```
#Count the empty (NaN, NAN, na) values in each column
df.isna().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
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [6]:

```
#Another check for any null / missing values
df.isnull().values.any()
```

Out[6]:

```
False
```

In [7]:

```
#View some basic statistical details like percentile, mean, standard deviation etc.
df.describe()
```

Out[7]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumbe
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.00000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86530
std	9.135373	403.509100	8.106864	1.024165	0.0	602.02433
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.25000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.50000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.75000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.00000

8 rows × 26 columns

In [8]:

```
#Get a count of the number of employee attrition, the number of employees that staye
df['Attrition'].value_counts()
```

Out[8]:

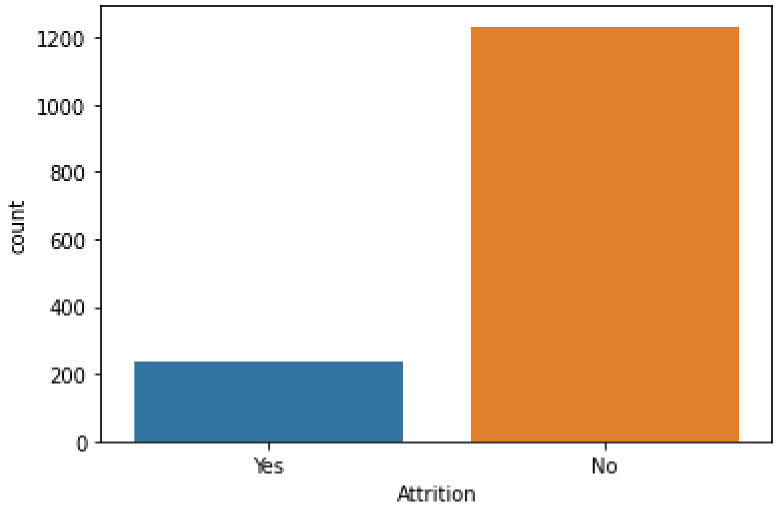
```
No     1233
Yes     237
```

```
Name: Attrition, dtype: int64
```

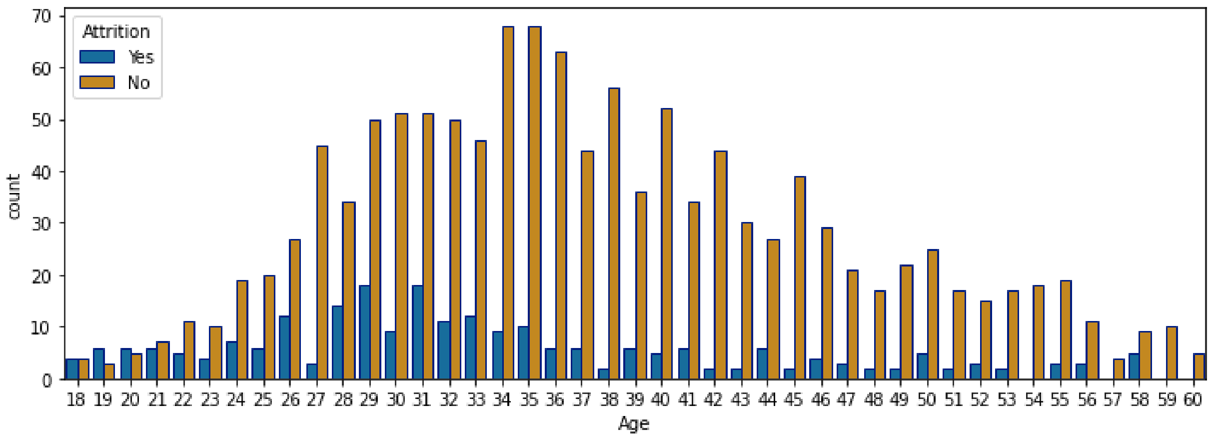
In [9]:
```
#Visualize this count
sns.countplot(df['Attrition'])
```

```
c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\seaborn\_decorators.py:43: 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.
  FutureWarning
```

Out[9]: `<AxesSubplot:xlabel='Attrition', ylabel='count'>`



In [10]:
```
#Show the number of employees that left and stayed by age
import matplotlib.pyplot as plt
fig_dims = (12, 4)
fig, ax = plt.subplots(figsize=fig_dims)

#ax = axis
sns.countplot(x='Age', hue='Attrition', data = df, palette="colorblind", ax = ax,  e
```



In [11]:
```
#Print all of the object data types and their unique values
for column in df.columns:
    if df[column].dtype == object:
        print(str(column) + ' : ' + str(df[column].unique()))
        print(df[column].value_counts())
        print("_____________________________________________________________________")
```

```
Attrition : ['Yes' 'No']
No      1233
Yes      237
```

```
Name: Attrition, dtype: int64
________________________________________________________________
BusinessTravel : ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
________________________________________________________________
Department : ['Sales' 'Research & Development' 'Human Resources']
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
________________________________________________________________
EducationField : ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
________________________________________________________________
Gender : ['Female' 'Male']
Male      882
Female    588
Name: Gender, dtype: int64
________________________________________________________________
JobRole : ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
________________________________________________________________
MaritalStatus : ['Single' 'Married' 'Divorced']
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
________________________________________________________________
Over18 : ['Y']
Y    1470
Name: Over18, dtype: int64
________________________________________________________________
OverTime : ['Yes' 'No']
No     1054
Yes     416
Name: OverTime, dtype: int64
________________________________________________________________
```

In [12]:

```
#Remove unneeded columns

#Remove the column EmployeeNumber
df = df.drop('EmployeeNumber', axis = 1) # A number assignment
#Remove the column StandardHours
df = df.drop('StandardHours', axis = 1) #Contains only value 80
#Remove the column EmployeeCount
df = df.drop('EmployeeCount', axis = 1) #Contains only the value 1
```

```
#Remove the column EmployeeCount
df = df.drop('Over18', axis = 1) #Contains only the value 'Yes'
```

In [13]:

```
#Get the correlation of the columns
df.corr()
```

Out[13]:

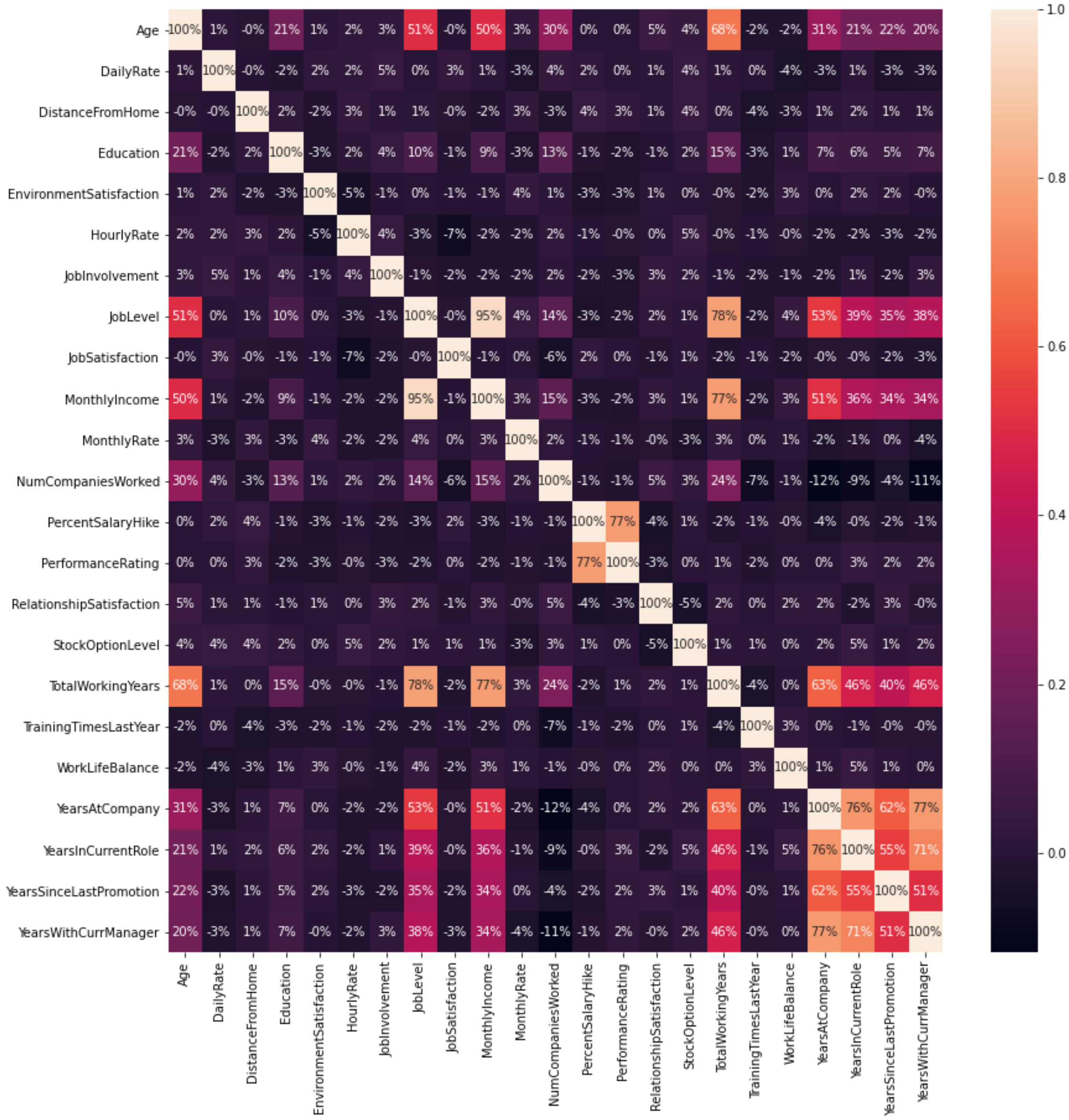
	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfacti
Age	1.000000	0.010661	-0.001686	0.208034	0.0101
DailyRate	0.010661	1.000000	-0.004985	-0.016806	0.0183
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	-0.0160
Education	0.208034	-0.016806	0.021042	1.000000	-0.0271
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	1.0000
HourlyRate	0.024287	0.023381	0.031131	0.016775	-0.0498
JobInvolvement	0.029820	0.046135	0.008783	0.042438	-0.0082
JobLevel	0.509604	0.002966	0.005303	0.101589	0.0012
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	-0.0067
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	-0.0062
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	0.0376
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	0.0125
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	-0.0317
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	-0.0295
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	0.0076
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	0.0034
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	-0.0026
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	-0.0193
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	0.0276
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	0.0014
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	0.0180
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	0.0161
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	-0.0049

23 rows × 23 columns

In [14]:

```
#Visualize the correlation
plt.figure(figsize=(14,14))  #14in by 14in
sns.heatmap(df.corr(), annot=True, fmt='.0%')
```

Out[14]: <AxesSubplot:>



In [15]:

```
#Check the structure of dataset
df.dtypes
```

Out[15]:

```
Age                          int64
Attrition                   object
BusinessTravel              object
DailyRate                    int64
Department                  object
DistanceFromHome             int64
Education                    int64
EducationField              object
EnvironmentSatisfaction      int64
Gender                      object
HourlyRate                   int64
JobInvolvement               int64
JobLevel                     int64
JobRole                     object
JobSatisfaction              int64
MaritalStatus               object
MonthlyIncome                int64
MonthlyRate                  int64
NumCompaniesWorked           int64
OverTime                    object
PercentSalaryHike            int64
```

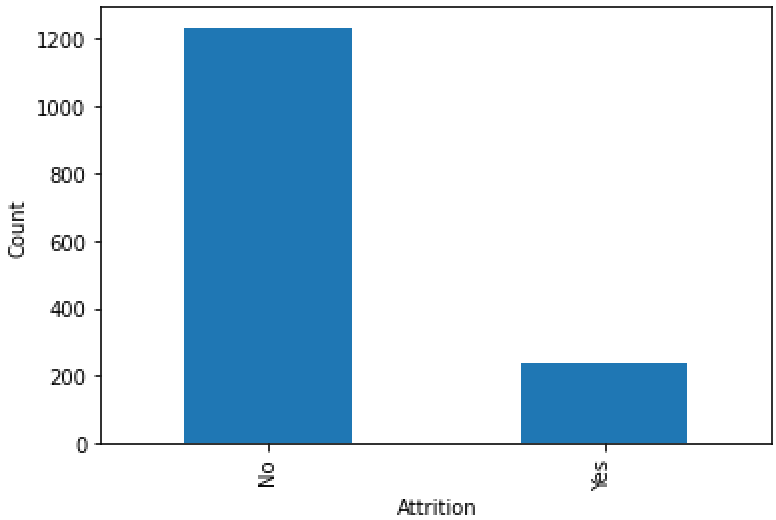
```
PerformanceRating           int64
RelationshipSatisfaction    int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

In [16]:

```
from sklearn.model_selection import train_test_split

#for fitting classification tree
from sklearn.tree import DecisionTreeClassifier

#to create a confusion matrix
from sklearn.metrics import confusion_matrix

#import whole class of metrics
from sklearn import metrics
```

In [17]:

```
df.Attrition.value_counts().plot(kind = "bar")
plt.xlabel("Attrition")
plt.ylabel("Count")
plt.show()
```



In [18]:

```
df["Attrition"].value_counts()
```

Out[18]:

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

In [19]:

```
#Transform non-numeric columns into numerical columns
from sklearn.preprocessing import LabelEncoder

for column in df.columns:
        if df[column].dtype == np.number:
            continue
        df[column] = LabelEncoder().fit_transform(df[column])
```

```
c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\ipykernel_launcher.py:5: Deprec
ationWarning: Converting `np.inexact` or `np.floating` to a dtype is deprecated. The
```

```
current result is `float64` which is not strictly correct.
  """
```

In [20]:

```
#Separating Feature and Target matrices
X = df.drop(['Attrition'], axis=1)
y=df['Attrition']
```

In [21]:

```
#Feature scaling is a method used to standardize the range of independent variables
#Since the range of values of raw data varies widely, in some machine learning algor
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
X = scale.fit_transform(X)
```

In [22]:

```
# Split the data into Training set and Testing set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.2,random_state=
```

In [25]:

```
#Function to plot Confusion Matrix
def cm_plot(cm,Model):
    plt.clf()
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Wistia)
    classNames = ['Negative','Positive']
    plt.title('Comparison of Prediction Result for '+ Model)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    tick_marks = np.arange(len(classNames))
    plt.xticks(tick_marks, classNames, rotation=45)
    plt.yticks(tick_marks, classNames)
    s = [['TN','FP'], ['FN', 'TP']]
    for i in range(2):
        for j in range(2):
            plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]))
    plt.show()
```

In [26]:

```
#Function to Train and Test Machine Learning Model
def train_test_ml_model(X_train,y_train,X_test,Model):
    model.fit(X_train,y_train) #Train the Model
    y_pred = model.predict(X_test) #Use the Model for prediction

    # Test the Model
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test,y_pred)
    accuracy = round(100*np.trace(cm)/np.sum(cm),1)

    #Plot/Display the results
    cm_plot(cm,Model)
    print('Accuracy of the Model' ,Model, str(accuracy)+'%')
```

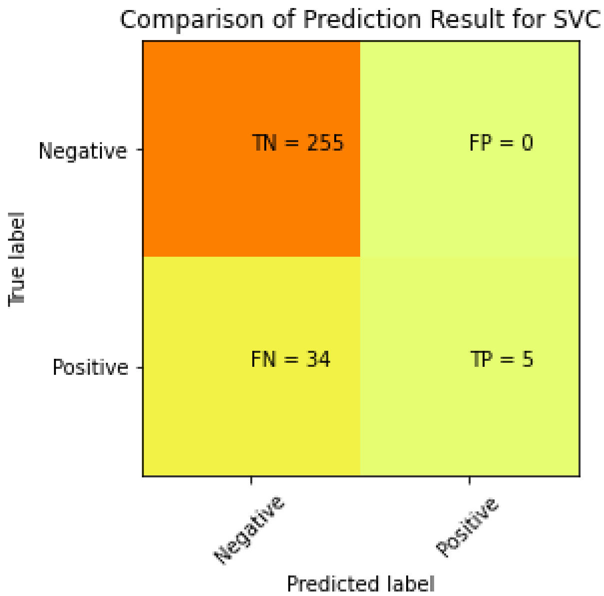
MACHINE LEARNING ALGORITHMS

1. SVC(Support Vector Classifier)

In [27]:

```
from sklearn.svm import SVC  #Import packages related to Model
Model = "SVC"
model=SVC() #Create the Model
```

```
train_test_ml_model(X_train,y_train,X_test,Model)
```

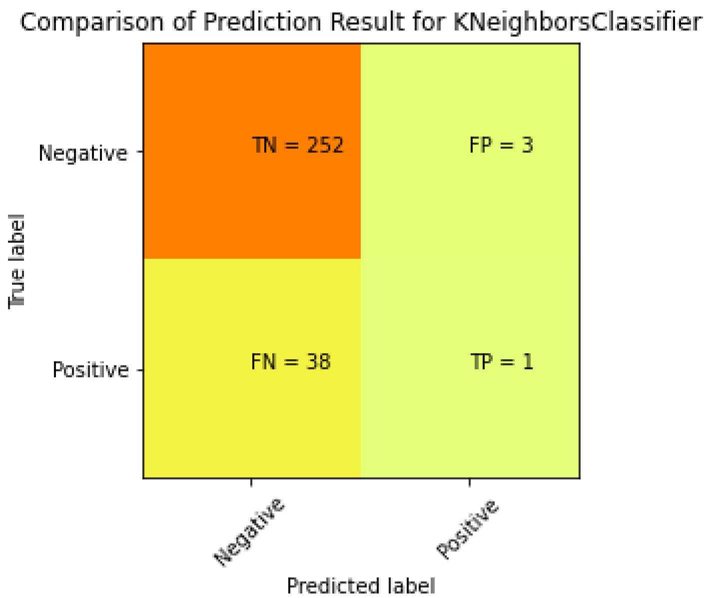



```
Accuracy of the Model SVC 88.4%
```

2. KNN CLASSIFIER

In [28]:

```
from sklearn.neighbors import KNeighborsClassifier  #Import packages related to Mode
Model = "KNeighborsClassifier"
model=KNeighborsClassifier()

train_test_ml_model(X_train,y_train,X_test,Model)
```

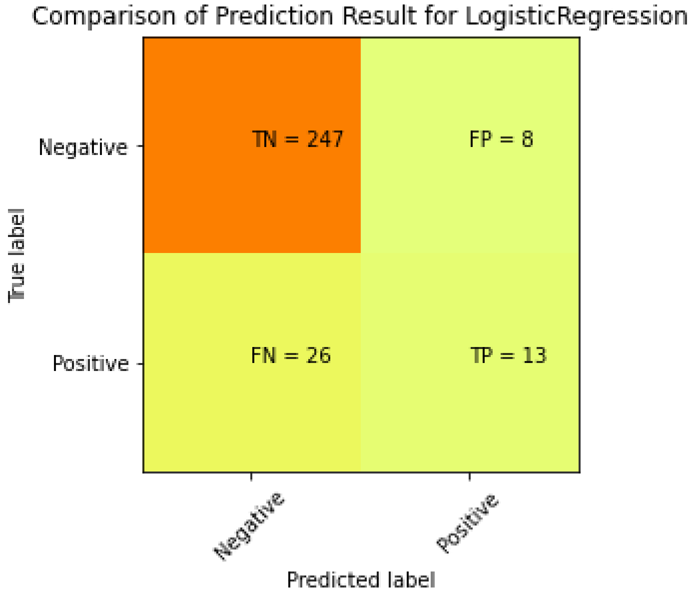



```
Accuracy of the Model KNeighborsClassifier 86.1%
```

3. LOGISTIC REGRESSIONS

In [29]:

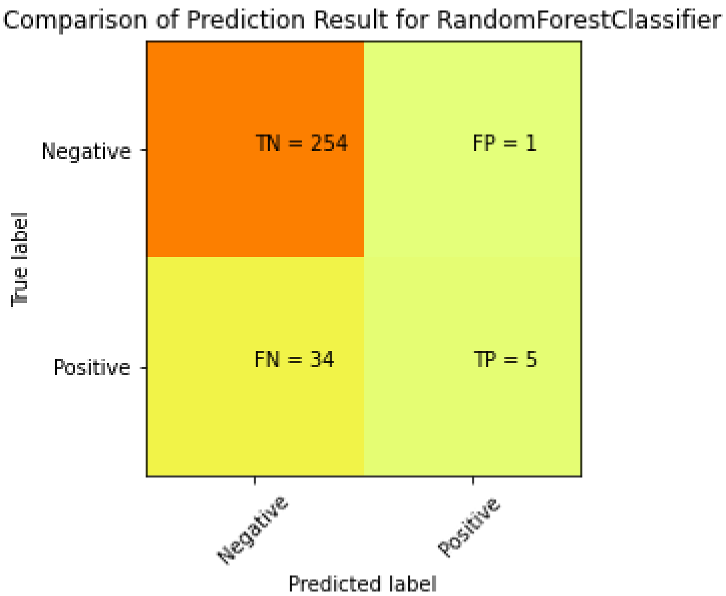
```
from sklearn.linear_model import LogisticRegression #Import packages related to Mode
Model = "LogisticRegression"
model=LogisticRegression()

train_test_ml_model(X_train,y_train,X_test,Model)
```




```
Accuracy of the Model LogisticRegression 88.4%
```

4. RANDOM FOREST

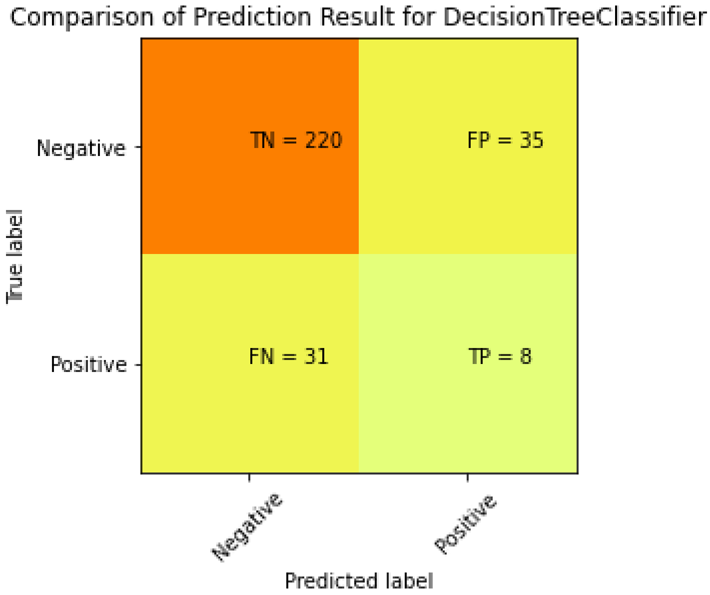
In [30]:
```
from sklearn.ensemble import RandomForestClassifier #Import packages related to Mode
Model = "RandomForestClassifier"
model=RandomForestClassifier()

train_test_ml_model(X_train,y_train,X_test,Model)
```




```
Accuracy of the Model RandomForestClassifier 88.1%
```

5. DECISION TREE

In [31]:
```
from sklearn.tree import DecisionTreeClassifier #Import packages related to Model
Model = "DecisionTreeClassifier"
model=DecisionTreeClassifier()

train_test_ml_model(X_train,y_train,X_test,Model)
```




```
Accuracy of the Model DecisionTreeClassifier 77.6%
```