DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Experiment No. 2.1

Student Name: Aayush Gurung **UID: 20BCS5323**
Branch: BE-CSE **Section/Group: 607/A**
Semester: 5th **Subject: Machine Learning Lab**

Aim: Classifying data using Support Vector Machines(SVMs) and Logistics Regression.

Software/Hardware Requirements: Windows 7 & above version

Tools to be done: Import any dataset and create data prediction table and graphs using SVM.

Output:

jupyter Untitled11 Last Checkpoint: 27 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [42]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

In [9]:
```
df=pd.read_csv(r"C:\Users\CU\Desktop\diabetes.csv")
```

In [10]:
```
df.head(10)
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

In [12]:
```
df.describe
```

Out[12]:
```
<bound method NDFrame.describe of      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

In [13]:
```
df.isnull().values.any()
```

Out[13]:
```
False
```

In [14]:
```
top_age = df.Age.value_counts().head(15)
top_age
```

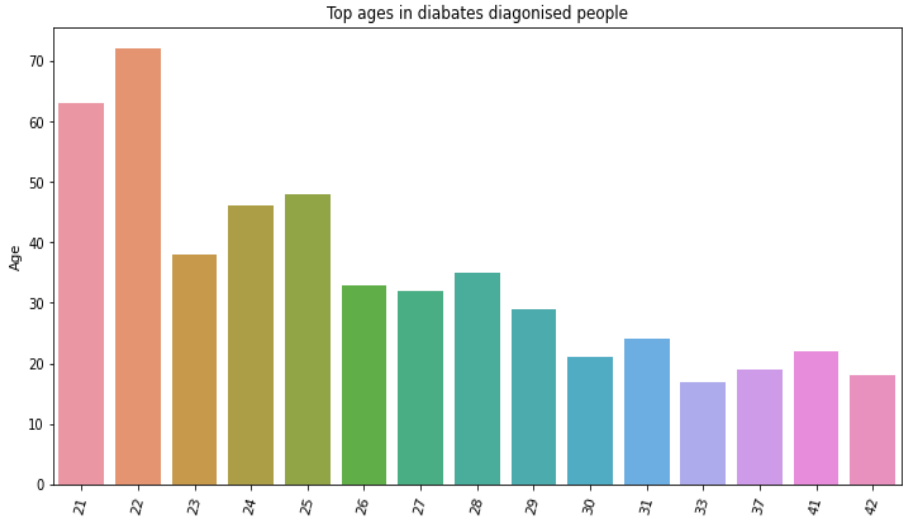
Out[14]:
```
22    72
21    63
25    48
24    46
23    38
28    35
26    33
27    32
29    29
31    24
41    22
30    21
37    19
42    18
33    17
Name: Age, dtype: int64
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [16]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=75)
plt.title('Top ages in diabates diagonised people')
sns.barplot(x=top_age.index, y=top_age)
```

Out[16]:
```
<AxesSubplot:title={'center':'Top ages in diabates diagonised people'}, ylabel='Age'>
```



In [17]:
```
y=df.drop(df.iloc[:,0:-1],axis=1)
```

In [18]:
```
y
```

In [18]:
```
y
```

Out[18]:

	Outcome
0	1
1	0
2	1
3	0
4	1
...	...
763	0
764	0
765	0
766	1
767	0

768 rows × 1 columns

In [19]:
```
x=df.iloc[:,:-1]
```

In [20]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y, test_size=0.25,random_state=7)
```

In [21]:
```
sv=SVC(probability=True)
```

In [22]:
```
sv.fit(xtrain,ytrain)
```

```
C:\Users\CU\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[22]:
```
SVC(probability=True)
```

In [23]:
```
pred4=sv.predict(xtest)
```

In [24]:
```
pred4
```

Out[24]:
```
array([0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

In [27]:
```
print("Accuracy:",metrics.accuracy_score(ytest, pred4))
print("Precision:",metrics.precision_score(ytest, pred4))
print("Recall:",metrics.recall_score(ytest, pred4))
```

```
Accuracy: 0.7552083333333334
Precision: 0.7804878048780488
Recall: 0.4571428571428713
```

In [32]:
```
logreg=LogisticRegression()
```

In [33]:
```
logreg.fit(xtrain,ytrain)
```

```
C:\Users\CU\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\CU\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (st
atus=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
  n_iter_i = _check_optimize_result(
```

Out[33]:
```
LogisticRegression()
```

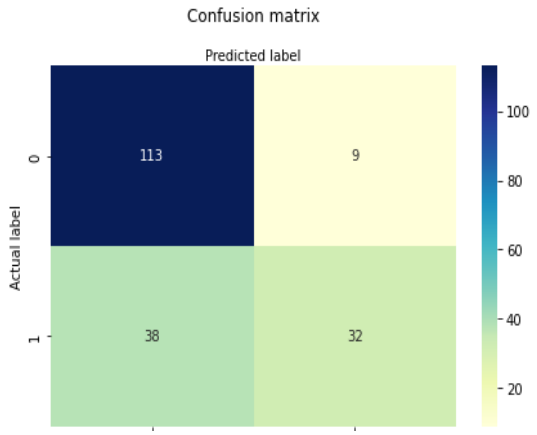
In [34]:
```
pred2=logreg.predict(xtest)
```

In [35]:
```
pred2
```

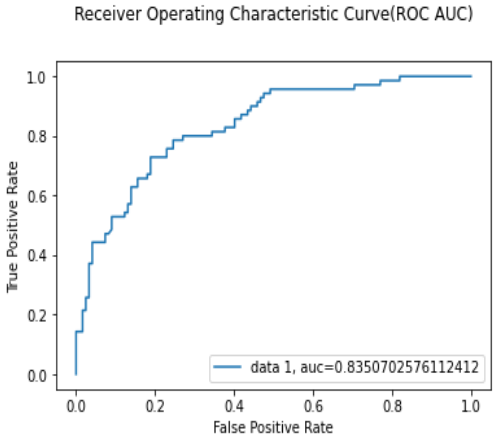
Out[35]:
```
array([0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

In [36]:
```
cnf_matrix = metrics.confusion_matrix(ytest, pred4)
cnf_matrix
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[36]: Text(0.5, 257.44, 'Predicted label')



In [37]:
```
y_pred_proba = sv.predict_proba(xtest)[::,1]
fpr, tpr, _ = metrics.roc_curve(ytest,  y_pred_proba)
auc = metrics.roc_auc_score(ytest, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.title('Receiver Operating Characteristic Curve(ROC AUC)', y=1.1)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



In [47]:
```
from sklearn.metrics import average_precision_score, precision_recall_curve
from sklearn.metrics import auc, plot_precision_recall_curve
```

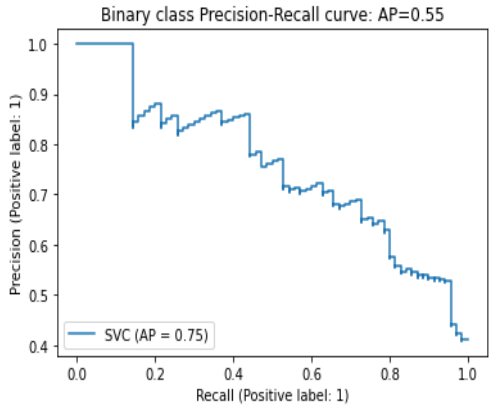
```
In [47]: from sklearn.metrics import average_precision_score, precision_recall_curve
         from sklearn.metrics import auc, plot_precision_recall_curve

In [49]: average_precision = average_precision_score(ytest, pred4)
         print(average_precision)
         disp = plot_precision_recall_curve(sv, xtest, ytest)
         disp.ax_.set_title('Binary class Precision-Recall curve: '
                            'AP={0:0.2f}'.format(average_precision))

0.5547110917537748

C:\Users\CU\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_precision_recall_curve is
deprecated; Function `plot_precision_recall_curve` is deprecated in 1.0 and will be removed in 1.2. Use one of the class method
s: PrecisionRecallDisplay.from_predictions or PrecisionRecallDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)

Out[49]: Text(0.5, 1.0, 'Binary class Precision-Recall curve: AP=0.55')
```



Learning outcomes (What I have learnt):

1. Understanding of logistic regression.
2. Able to implement logistic regression on any given dataset.
3. Learning about different library/packages of python.
4. Learning of different Machine Learning Functions

Evaluation Grid :

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Student Performance (Conduct of experiment) objectives/Outcomes.		12
2.	Viva Voce		10
3.	Submission of Work Sheet (Record)		8
	Total		30