Pabna University of Science & Technology

Department of

Information and Communication Engineering

Faculty of Engineering and Technology

Assignment on

SVM Algorithm

Course Title: Neural Networks Sessional

Course Code: ICE-4206

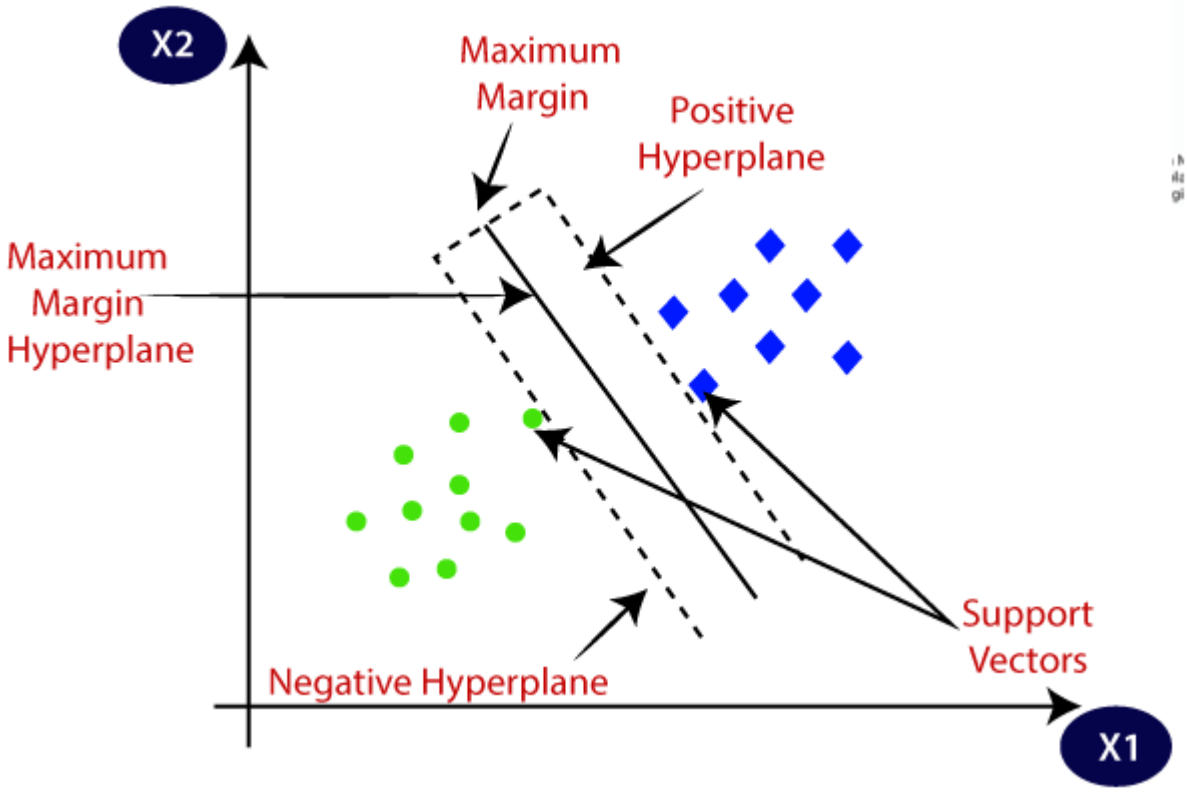
Submitted By: **Roll: 190617, 190625, 190638** **Session: 2018-2019** **4th Year 2nd Semester,** **Department of ICE,** **Pabna University of Science and Technology.** **Submission Date: 17-08-2024**	**Submitted To:** **Dr. Md. Imran Hossain** **Associate Professor** **Department of ICE,** **Pabna University of Science and Technology.** **Teacher's Signature:**

Support Vector Machine Algorithm (SVM):

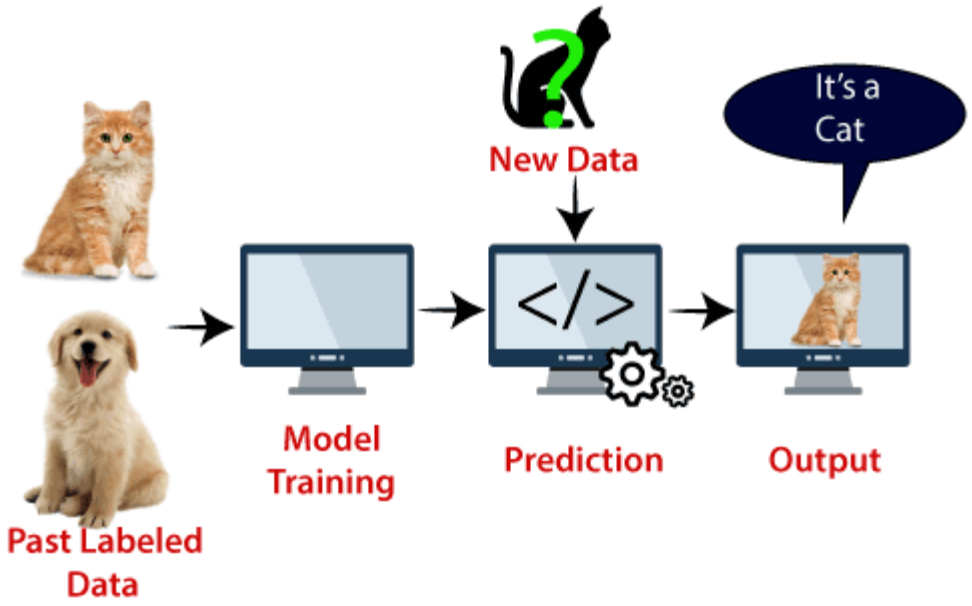
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

a) **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

b) **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

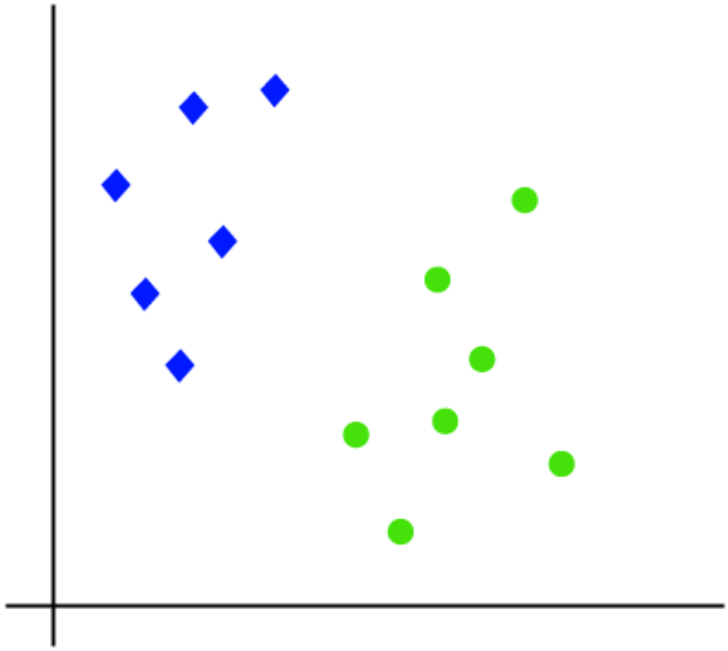
We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

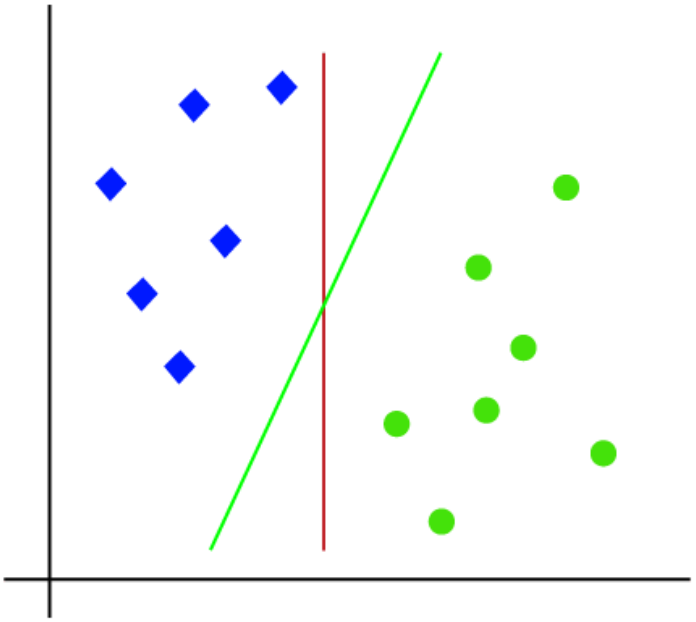
How does SVM works?

Linear SVM:

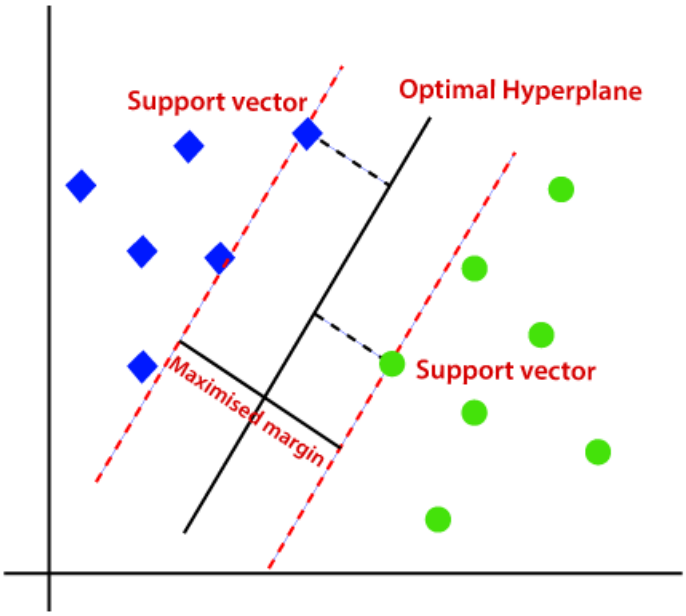
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

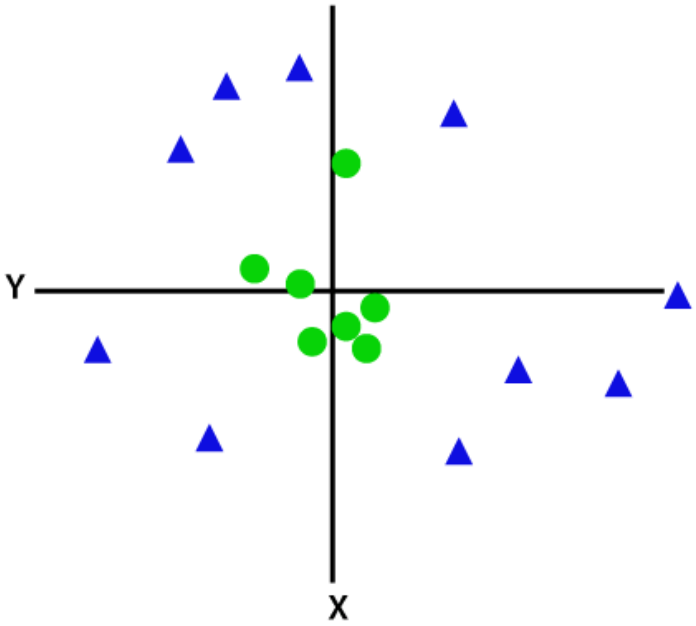


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

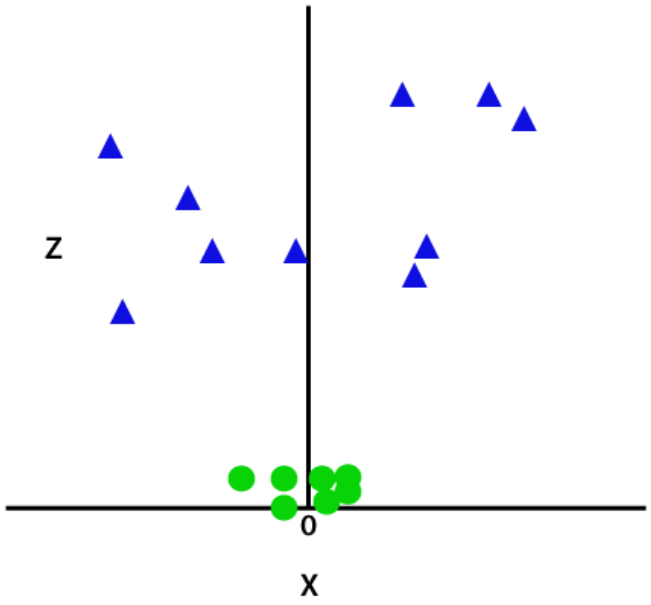
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



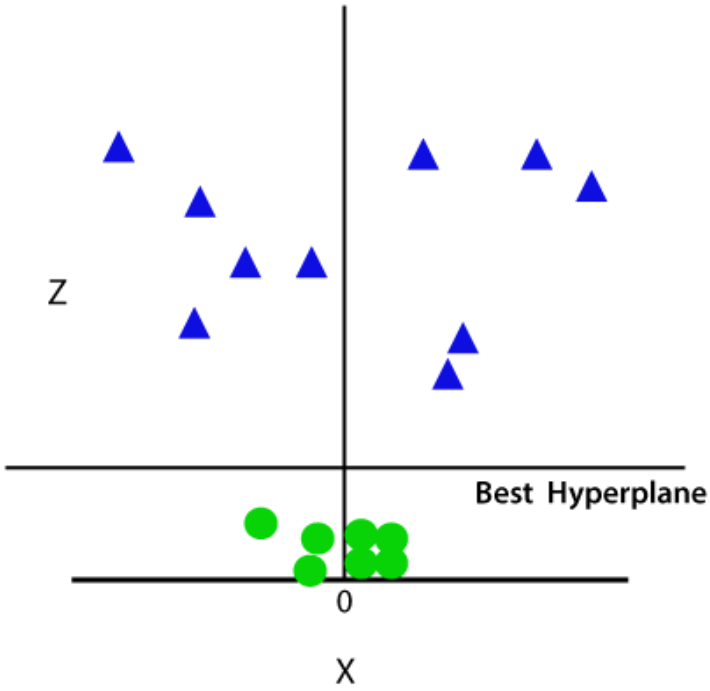
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z= x^2 +y^2$

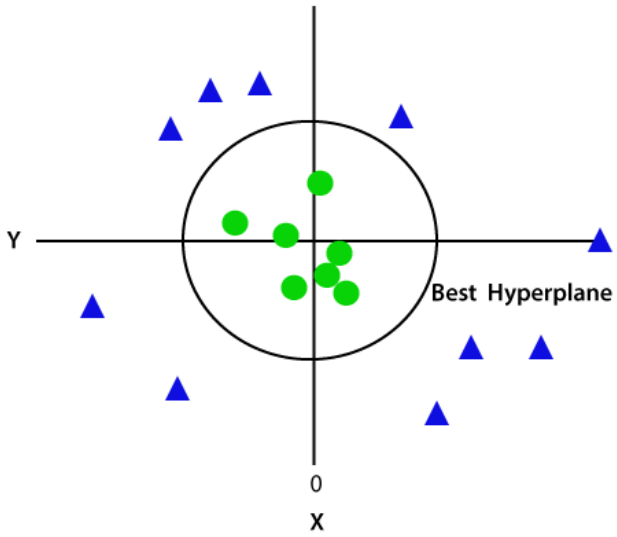
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

Project: **User Purchase Classification Prediction using SVM**

Project Description: This project implements the Support Vector Machine (SVM) algorithm for predicting user purchase classification. It utilizes the user-data.csv dataset, which contains information about users and their purchase behavior. The goal is to train an SVM classifier to predict whether a user will purchase a particular product or not.

Data Pre-processing Step

The first step in the project is data pre-processing. The user_data.csv file is imported using the Pandas library, and the independent variables (features) and the dependent variable (target) are extracted from the dataset. The dataset is then split into training and test sets using the train_test_split function from the sklearn.model_selection module. Feature scaling is performed using the StandardScaler class from the sklearn.preprocessing module.

SVM Algorithm

The SVM algorithm is implemented using the SVC (Support Vector Classifier) class from the sklearn.svm library. In this project, a linear kernel is used to create the SVM classifier for linearly separable data. The classifier is trained on the training set using the fit method.

Predicting the Test Set Result

The trained SVM classifier is used to predict the output for the test set. The predict method is used to obtain the predicted values, which are stored in the y_pred vector. The predicted values can be compared with the actual values (y_test) to evaluate the performance of the classifier.

Confusion Matrix

The performance of the SVM classifier is evaluated by creating a confusion matrix. The confusion_matrix function from the sklearn.metrics module is used to generate the confusion matrix. It takes the actual values (y_test) and the predicted values (y_pred) as parameters.

Visualizing the Training Set Result

The training set result is visualized using the contourf and scatter functions from the matplotlib.pyplot module. The contourf function is used to create a filled contour plot, representing the decision boundary of the SVM classifier. The scatter function is used to plot the data points, with different colors representing different classes. The plot is labeled with appropriate titles, axis labels, and legends.

Visualizing the Test Set Result

The test set result is visualized in a similar manner as the training set result. The contourf and scatter functions are used to create the filled contour plot and plot the data points, respectively. The plot is labeled with appropriate titles, axis labels, and legends.

Conclusion

The SVM classifier successfully predicts user purchase classification based on the provided dataset. The project demonstrates the use of SVM algorithm for binary classification tasks and provides visualizations of the decision boundary and class separation.

Source Code & Output:

```
from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

from matplotlib.colors import ListedColormap

import seaborn as sns

# Importing data and creating a dataframe.
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/new-dataset/user-data.csv")
display(df.head(20))
display(df.dtypes)
```

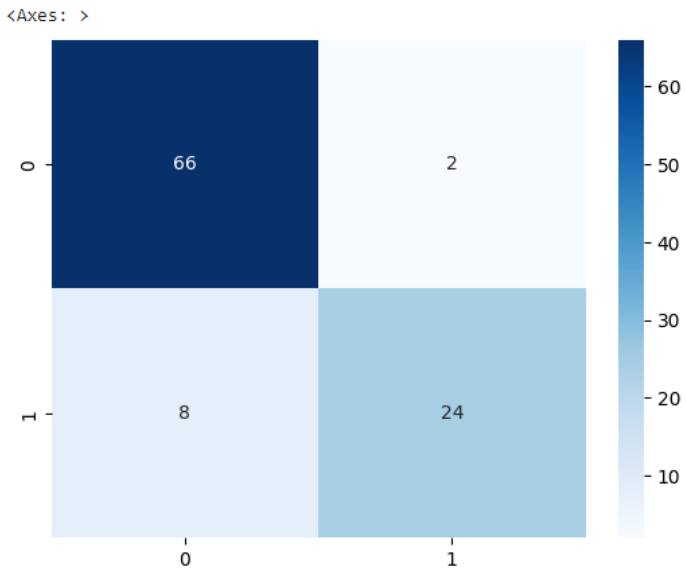
	user_id	gender	age	estimated_salary	purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

	0
user_id	int64
gender	object
age	int64
estimated_salary	int64
purchased	int64

dtype: object

```
# Extracting independent and dependent variables.
x = df.iloc[:, [2, 3]].values
y = df.iloc[:, 4].values

# Output first five values in both the lists.
print(x[:10])
```

```
print(y[:10])

# Splitting the data into training and testing data.
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
std_x = StandardScaler()
x_train = std_x.fit_transform(x_train)
x_test = std_x.transform(x_test)
print("x_train:", x_train[:5], "...")
print("")
print("x_test:", x_test[:5], "...")
model = SVC(kernel="linear", random_state=0)
model.fit(x_train, y_train)

# Predicting the testing dataset results
y_pred = model.predict(x_test)
print(y_pred[:10], "...")
[0 0 0 0 0 0 0 1 0 0] ...
df = pd.DataFrame(y_test, y_pred)
df.head(-5)

#Creating the Confusion matrix and heatmap
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
sns.heatmap(cm, annot=True, cmap="Blues")
```




```
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
```

Accuracy: 0.9

```
from sklearn.metrics import classification_report
```

```
# Generate predictions for the test set
y_pred = model.predict(x_test)

# Generate and display the classification report
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.89      0.97      0.93        68
           1       0.92      0.75      0.83        32

    accuracy                           0.90       100
   macro avg       0.91      0.86      0.88       100
weighted avg       0.90      0.90      0.90       100
```

```
def check_classification(input_feature_1, input_feature_2):
  # Create an input array using the two feature values
  input_data = np.array([[input_feature_1, input_feature_2]])

  # Predict the class using the trained model
  predicted_class = model.predict(input_data)

  return predicted_class[0]

# Example usage:
# Replace the input values with the specific feature values you want to test
result = check_classification(input_feature_1=32 , input_feature_2=150000)
print(f"The predicted class is: {result}")
```

```
The predicted class is: 1
```

```
##This is code for SVM classifier traning set

x_set, y_set = x_train, y_train

x1, x2 = np.meshgrid(np.arange(start=x_set[:, 0].min() - 1, stop=x_set[:, 0].max() + 1, step=0.01),
np.arange(start=x_set[:, 1].min() - 1, stop=x_set[:, 1].max() + 1, step=0.01))

plt.contourf(x1, x2, model.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape), alpha = 0.75,
cmap = ListedColormap(('orange', 'dodgerblue')))
plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y_set)):
  plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c = [ListedColormap(('red', 'white'))(i)], label = j)

plt.title('SVM classifier (Training set)')

plt.xlabel('Age')
```

```
plt.ylabel('Estimated Salary')

plt.legend()

plt.show()
```