Experiment:- 4

Student Name: Ruchika Raj **UID: 20BCS9285**
Branch: CSE **Section/Group: 20BCS_WM_615/B**
Semester: 5th **Date of Performance: 26/09/2022**
Subject Code: 20CSP-317
Subject Name: MACHINE LEARNING LAB

Aim/Overview of the practical:

Classifying data using Support Vector Machines(SVMs) in Python

Task to be done:

To perform Classification using Support Vector Machines(SVMs) on any standard dataset.

Apparatus/Simulator used:

- Jupyter Notebook/Google Collab
- Python
- pandas Library
- seaborn Library
- Standard Dataset

Code and Output:

```
#importing necessary libraries
import numpy as np import
pandas as pd import
matplotlib import
matplotlib.pyplot as plt from
sklearn import svm

#Import Dataset data1 =
pd.read_csv('Social_Network_Ads.csv')
data1.head()

data1.corr()

X = data1.iloc[:,[2,3]].values y
= data1.iloc[:,4].values

#Splitting dataset to test and train set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=0)

#Feature Scaling from sklearn.preprocessing
import StandardScaler sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Create a model for SVM from sklearn.svm
import SVC classifier =
SVC(kernel='linear',random_state=0)
```

#classifier = SVC(kernel='poly',degree=5,random_state=0) //Poly kernel will predict less accurate since our data is linear. We use Linear kernel for SVM classifier.fit(X_train,y_train)

# get prediction values for train data from sklearn.model_selection
import cross_val_score, cross_val_predict lModel = svm.SVC(kernel =
'linear') lModel.fit(X_train,y_train)
Y_pred = cross_val_predict(lModel,X_train,y_train,cv =3)
Y_pred

#Prediction of test data set y_pred
= classifier.predict(X_test) y_pred

#Making the confusion matrix from sklearn.metrics import confusion_matrix,
precision_score, recall_score, f1_score cm = confusion_matrix(y_test,y_pred) cm

plt.scatter(X_train[:,0], X_train[:,1],c=y_train) plt.show

plt.scatter(X_test[:,0], X_test[:,1],c=y_test) plt.show

instantiate model with kernel = 'linear' lModel
= svm.SVC(kernel = 'linear')
lModel.fit(X_train,y_train)
# cross validation to get avg accuracy and std
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict

score = cross_val_score(lModel,X_train,y_train,cv =10, scoring = 'accuracy')
print("avg accuracy:\t{0:,.4f}".format(np.mean(score))) print("avg
std:\t{0:,.4f}".format(np.std(score)))

jupyter SVM EXP Last Checkpoint: 12 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [1]:
```
#importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from sklearn import svm
```

In [2]:
```
#Import Dataset
data1 = pd.read_csv('Social_Network_Ads.csv')
data1.head()
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	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

In [3]:
```
data1.corr()
```

Out[3]:

	User ID	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.000721	0.071097	0.007120
Age	-0.000721	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	0.155238	1.000000	0.362083
Purchased	0.007120	0.622454	0.362083	1.000000

jupyter SVM EXP Last Checkpoint: 12 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

EstimatedSalary	0.071097	0.155238	1.000000	0.362083
Purchased	0.007120	0.622454	0.362083	1.000000

In [4]:
```
X = data1.iloc[:,[2,3]].values
y = data1.iloc[:,4].values
```

In [5]:
```
#Splitting dataset to test and train set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=0)
```

In [6]:
```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [7]:
```
#Create a model for SVM
from sklearn.svm import SVC
classifier = SVC(kernel='linear',random_state=0)
#classifier = SVC(kernel='poly',degree=5,random_state=0) //Poly kernel will predict less accurate since our data is linear. We us
classifier.fit(X_train,y_train)
```

Out[7]:
```
SVC(kernel='linear', random_state=0)
```

In [8]:
```
 # get prediction values for train data
from sklearn.model_selection import cross_val_score, cross_val_predict
lModel = svm.SVC(kernel = 'linear')
lModel.fit(X_train,y_train)
Y_pred = cross_val_predict(lModel,X_train,y_train,cv =3)
Y_pred
```

jupyter SVM EXP Last Checkpoint: 12 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

```
Y_pred = cross_val_predict(imodel,X_train,y_train,cv =3)
Y_pred
```

Out[8]:
```
array([0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [9]:
```
#Prediction of  test data set
y_pred = classifier.predict(X_test)
y_pred
```

Out[9]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

In [10]:
```
#Making the confusion matrix
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[10]:
```
array([[66,  2],
```

jupyter SVM EXP Last Checkpoint: 12 hours ago (autosaved) Logout

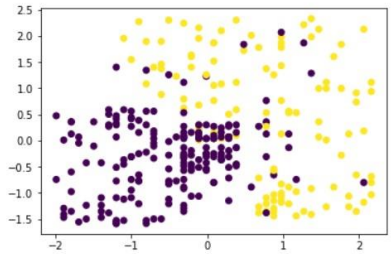
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
       [ 8, 24]], dtype=int64)
```

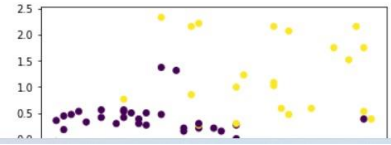
In [11]:
```
plt.scatter(X_train[:,0], X_train[:,1],c=y_train)
plt.show
```

Out[11]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [12]:
```
plt.scatter(X_test[:,0], X_test[:,1],c=y_test)
plt.show
```

Out[12]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

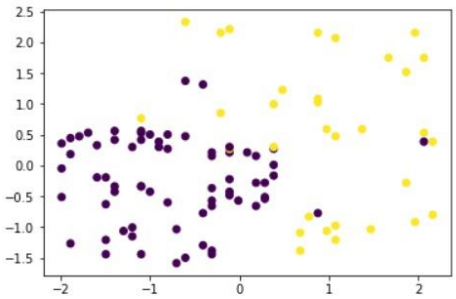


jupyter SVM EXP Last Checkpoint: 12 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [12]:
```
plt.scatter(X_test[:,0], X_test[:,1],c=y_test)
plt.show
```

Out[12]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [13]:
```
# instantiate model with kernel = 'linear'
lModel = svm.SVC(kernel = 'linear')
lModel.fit(X_train,y_train)
# cross validation to get avg accuracy and std
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict

score = cross_val_score(lModel,X_train,y_train,cv =10, scoring = 'accuracy' )
print("avg accuracy:\t{0:,.4f}".format(np.mean(score)))
print("avg std:\t{0:,.4f}".format(np.std(score)))
```

```
avg accuracy:	0.8133
avg std:	0.0884
```

<u>Learning outcomes (What I have learnt)</u>:

1. To understand Support Vector Machines(SVMs)
2. Learn about pandas', matplotlib and seaborn library/package of python.
3. Learn about the different methods/functions that are needed to generate different types of graphs, charts and plots of the given dataset.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			