EXPERIMENT 6(b)

IMPLEMENTATION OF SVM KERNEL ALGORITHM

NAME : SREENIDHI GANACHARI

REGISTRATION NUMBER : 19BCE7230

SLOT : L-23 &24

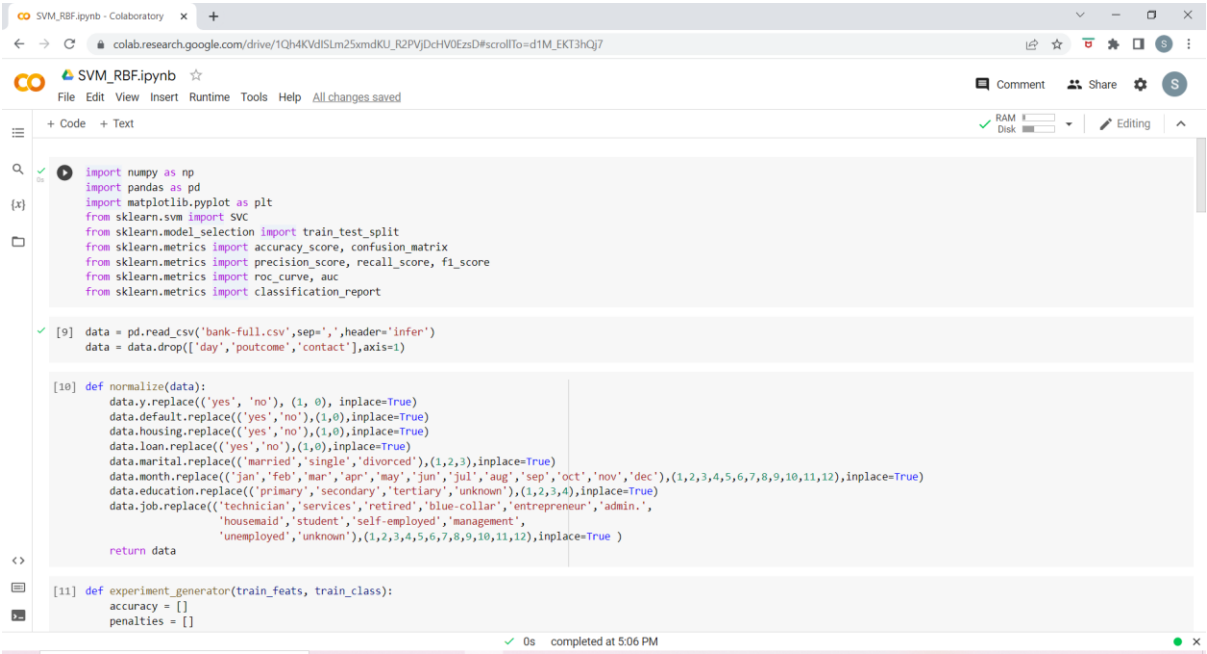
CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report
data = pd.read_csv('bank-full.csv',sep=',',header='infer')
data = data.drop(['day','poutcome','contact'],axis=1)
def normalize(data):
    data.y.replace(('yes', 'no'), (1, 0), inplace=True)
    data.default.replace(('yes','no'),(1,0),inplace=True)
    data.housing.replace(('yes','no'),(1,0),inplace=True)
    data.loan.replace(('yes','no'),(1,0),inplace=True)
    data.marital.replace(('married','single','divorced'),(1,2,3),inplac
e=True)
    data.month.replace(('jan','feb','mar','apr','may','jun','jul','aug'
,'sep','oct','nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12),inplace=True)
    data.education.replace(('primary','secondary','tertiary','unknown')
,(1,2,3,4),inplace=True)
    data.job.replace(('technician','services','retired','blue-
collar','entrepreneur','admin.',
                      'housemaid','student','self-
employed','management',
                      'unemployed','unknown'),(1,2,3,4,5,6,7,8,9,10,11,
12),inplace=True )
    return data
def experiment_generator(train_feats, train_class):
    accuracy = []
    penalties = []
    G = .00000001
    penalty = 1
    N = 10
    for item in range(N):
```

```
        clf = SVC(kernel='rbf', random_state = 0, gamma = G, C = penalt
y)
        clf.fit(train_feats, train_class.values.ravel())
        pred_train = clf.predict(train_feats)

        s_train = accuracy_score(train_class, pred_train)

        penalties.append(penalty)
        accuracy.append(s_train)

        penalty += 1
        G += .00000001
    plt.scatter(penalties, accuracy)
    plt.ylabel('Accuracy (%')
    plt.xlabel('Penalty - C Parameter')
    plt.show()



data = normalize(data)

plt.hist((data.duration),bins=100)
plt.ylabel('Occurences (Frequency)')
plt.xlabel('Client Call Duration')
plt.show()
plt.hist((data.job),bins=10)
plt.ylabel('Occurences (Frequency)')
plt.xlabel('Client Job Indices')
plt.show()
plt.hist((data.balance),bins=10)
plt.ylabel('Occurences (Frequency)')
plt.xlabel('Client Balance')
plt.show()
X_train, X_test, y_train, y_test = train_test_split(data, data.y, test_
size=0.2)
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)

df_train = X_train
df_test = X_test

df_train_class = pd.DataFrame(df_train['y'])
df_train_features = df_train.loc[:, df_train.columns != 'y']

df_test_class = pd.DataFrame(df_test['y'])
df_test_features = df_test.loc[:, df_test.columns != 'y']

g = .0001
c = 1
```

```
mlp_classifier = SVC(kernel='rbf', random_state = 0, gamma = g, C = c)
mlp_classifier.fit(df_train_features, df_train_class.values.ravel())

predicted_train = mlp_classifier.predict(df_train_features)
predicted_test = mlp_classifier.predict(df_test_features)

score_train = accuracy_score(df_train_class, predicted_train)
score_test = accuracy_score(df_test_class, predicted_test)

print('Training Accuracy Score: {}'.format(score_train))
print('Testing Accuracy Score: {}'.format(score_test)

precision_train = precision_score(df_train_class, predicted_train)
precision_test = precision_score(df_test_class, predicted_test)
print('Training Precision: {}'.format(precision_train))
print('Testing Precision: {}'.format(precision_test))

recall_train = recall_score(df_train_class, predicted_train)
recall_test = recall_score(df_test_class, predicted_test)
print('Training Recall: {}'.format(recall_train))
print('Testing Recall: {}'.format(recall_test))


print('Training Classification Report: ')
print(classification_report(df_train_class, predicted_train))
print('Testing Classification Report: ')
print(classification_report(df_test_class, predicted_test))
```

OUTPUT -



SVM_RBF.ipynb - Colaboratory

colab.research.google.com/drive/1Qh4KVdISLm25xmdKU_R2PVjDcHV0EzsD#scrollTo=d1M_EKT3hQj7

SVM_RBF.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
def experiment_generator(train_feats, train_class):
    accuracy = []
    penalties = []
    G = .00000001
    penalty = 1
    N = 10
    for item in range(N):
        clf = SVC(kernel='rbf', random_state = 0, gamma = G, C = penalty)
        clf.fit(train_feats, train_class.values.ravel())
        pred_train = clf.predict(train_feats)

        s_train = accuracy_score(train_class, pred_train)

        penalties.append(penalty)
        accuracy.append(s_train)

        penalty += 1
        G += .00000001
    plt.scatter(penalties, accuracy)
    plt.ylabel('Accuracy (%')
    plt.xlabel('Penalty - C Parameter')
    plt.show()


data = normalize(data)
```

```
[12] plt.hist((data.duration),bins=100)
     plt.ylabel('Occurences (Frequency)')
     plt.xlabel('Client Call Duration')
```

0s completed at 5:06 PM

SVM_RBF.ipynb - Colaboratory

colab.research.google.com/drive/1Qh4KVdISLm25xmdKU_R2PVjDcHV0EzsD#scrollTo=d1M_EKT3hQj7

SVM_RBF.ipynb

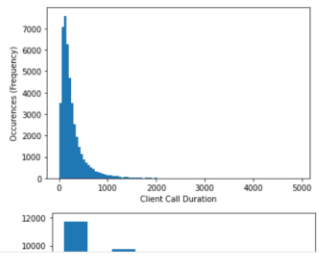
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
plt.hist((data.duration),bins=100)
plt.ylabel('Occurences (Frequency)')
plt.xlabel('Client Call Duration')
plt.show()
plt.hist((data.job),bins=10)
plt.ylabel('Occurences (Frequency)')
plt.xlabel('Client Job Indices')
plt.show()
plt.hist((data.balance),bins=10)
plt.ylabel('Occurences (Frequency)')
plt.xlabel('Client Balance')
plt.show()
```



0s completed at 5:06 PM

SVM_RBF.ipynb - Colaboratory

colab.research.google.com/drive/1Qh4KVdISLm25xmdKU_R2PVjDcHV0EzsD#scrollTo=d1M_EKT3hQj7

SVM_RBF.ipynb

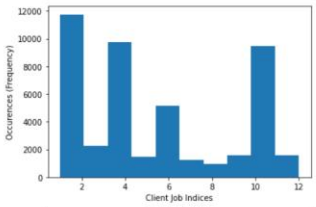
File Edit View Insert Runtime Tools Help All changes saved

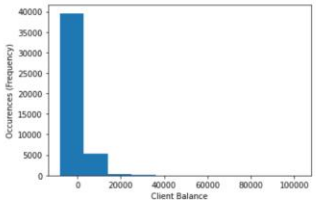
Comment Share

+ Code + Text

RAM Disk Editing

[12]






```
X_train, X_test, y_train, y_test = train_test_split(data, data.y, test_size=0.2)
print (X_train.shape, y_train.shape)
```

0s completed at 5:06 PM

SVM_RBF.ipynb - Colaboratory

colab.research.google.com/drive/1Qh4KVdISLm25xmdKU_R2PVjDcHV0EzsD#scrollTo=d1M_EKT3hQj7

SVM_RBF.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[13]

```
X_train, X_test, y_train, y_test = train_test_split(data, data.y, test_size=0.2)
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)

df_train = X_train
df_test = X_test

df_train_class = pd.DataFrame(df_train['y'])
df_train_features = df_train.loc[:, df_train.columns != 'y']

df_test_class = pd.DataFrame(df_test['y'])
df_test_features = df_test.loc[:, df_test.columns != 'y']

g = .0001
c = 1

mlp_classifier = SVC(kernel='rbf', random_state = 0, gamma = g, C = c)
mlp_classifier.fit(df_train_features, df_train_class.values.ravel())

predicted_train = mlp_classifier.predict(df_train_features)
predicted_test = mlp_classifier.predict(df_test_features)
```

```
(36168, 14) (36168,)
(9043, 14) (9043,)
```

```
score_train = accuracy_score(df_train_class, predicted_train)
score_test = accuracy_score(df_test_class, predicted_test)

print('Training Accuracy Score: {}'.format(score_train))
```

0s completed at 5:06 PM

SVM_RBF.ipynb - Colaboratory

colab.research.google.com/drive/1Qh4KVdISLm25xmdKU_R2PVjDcHV0EzsD#scrollTo=d1M_EKT3hQj7

SVM_RBF.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[14]

```
score_train = accuracy_score(df_train_class, predicted_train)
score_test = accuracy_score(df_test_class, predicted_test)

print('Training Accuracy Score: {}'.format(score_train))
print('Testing Accuracy Score: {}'.format(score_test))
```

```
Training Accuracy Score: 0.9121875691218757
Testing Accuracy Score: 0.8893066460245493
```

```
precision_train = precision_score(df_train_class, predicted_train)
precision_test = precision_score(df_test_class, predicted_test)
print('Training Precision: {}'.format(precision_train))
print('Testing Precision: {}'.format(precision_test))

recall_train = recall_score(df_train_class, predicted_train)
recall_test = recall_score(df_test_class, predicted_test)
print('Training Recall: {}'.format(recall_train))
print('Testing Recall: {}'.format(recall_test))
```

```
Training Precision: 0.802624073017684
Testing Precision: 0.5683646112600537
Training Recall: 0.3320745810715129
Testing Recall: 0.2015209125475285 2
```

0s completed at 5:06 PM

SVM_RBF.ipynb - Colaboratory

colab.research.google.com/drive/1Qh4KVdISLm25xmdKU_R2PVjDcHV0EzsD#scrollTo=d1M_EKT3hQj7

SVM_RBF.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
print('Training Classification Report: ')
print(classification_report(df_train_class, predicted_train))
print('Testing Classification Report: ')
print(classification_report(df_test_class, predicted_test))
```

```
Training Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.99      0.95     31931
           1       0.80      0.33      0.47      4237

    accuracy                           0.91     36168
   macro avg       0.86      0.66      0.71     36168
weighted avg       0.90      0.91      0.90     36168

Testing Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      7991
           1       0.57      0.20      0.30      1052

    accuracy                           0.89      9043
   macro avg       0.74      0.59      0.62      9043
weighted avg       0.86      0.89      0.87      9043
```

+ Code + Text

0s completed at 5:06 PM