Assignment 3

Data Set Generation and classifying into classes regions, splitting training/testing, Executing the Perceptron/Linear SVM/ Kernel SVM, fine tunning "C" Parameters and"Gamma" values to improve the accuracy between the predicted and calcuted test patterns.

Tabulated the coefficient, accuracy, weighted vectors and intercept along with

Plotting the accuracy graph/tables and Plotting the Datasets features are done here.

Dataset -

(a) Each pattern here is a two-dimensional vector X = x numbers in the range [–5, 5]. , where both x1 and x2 are real (b) A pair of linearly separable classes are C1 and C2 and are shown by one of the dotted lines specifying the boundary x1 = x2. They are given by C1 ={(x1,x2)t :x1 >x2}andC2 = {(x1,x2)t :x1 <x2}. (c) The second pair of linearly separable classes are C3 and C4 as shown by the other dotted line in the Figure. They are given by C3 ={(x1,x2)t :x1 +x2 +1>0}andC4 ={(x1,x2)t :x1 +x2 +1<0}. (d) So, these two pairs of classifiers generate four regions based on the four classes C1, C2, C3, and C4. These four regions are given by R13 ={X :X ∈C1 andX ∈C3}, R14 ={X :X ∈C1 andX ∈C4} and R24 ={X :X ∈C2 andX ∈C4}, R23 ={X :X ∈C2 andX ∈C3} (e) Now the two classes that are not lineraly separable are defined using these foure regions as NC1 ={X:X∈R13 orX∈R24},andNC2 = {X:X∈R14 orX∈R23}. So, NC1 is the union of R13 and R24 and similarlyNC2 is the union of R23 and R14.

X1 X2 X1Square X2Square Y C1 C2 C3 C4 R13 R14 R24 R23 NC1 NC2 NC-Class

0 2.60 -2.80 6.76 7.84 1 Yes No Yes No Yes No No No Yes No 1

1 3.40 3.30 11.56 10.89 1 Yes No Yes No Yes No No No Yes No 1

7 -0.50 -1.30 0.25 1.69 1 Yes No No Yes No Yes No No No Yes 0

The patterns are generated between the range[-5,5], classified in [C1.,C2,C3,C4, R31,R24,R14,R24,NC1,NC2] based the above below. If it satisfy the Class - it is marked YES else NO- similar for Regions and NC. Also NC are Labelled as [0,1] and for Kernel Algo - we have [Y] Label since we need to compute for2 Dimension.

Perceptron Linear SVM, Kernel SVM Data Set [200 300 400] Output

++++++++++++++++++++++++++SVM++++++++++++++++++++++

Weighted Vector :::: 0 200 {'Weight': array([[0.26, -0.29, 0.31, -0.38]]), 'intercept'*: array([0.02]), 'accuracy': 0.9833333333333333} Weighted Vector :::: 1 300 {'Weight': array([[0.01, -0.01, 0.07, -0.24]]), 'intercept*': array([1.05]), 'accuracy': 0.8166666666666667} Weighted Vector :::: 2 400 {'Weight': array([[0.03, -0.03, 0.15, -0.23]]), 'intercept_': array([0.31]), 'accuracy': 0.925}

+++++++++++++++++++++++++++++++Perceptron++++++++++++++++++

Weighted Vector :::: 200 {'Weight': array([[25.3 , -18.3 , 24.69, -30.27]]), 'intercept'*: array([5.]), 'accuracy': 0.975} Weighted Vector :::: 300 {'Weight': array([[5.5 , -1.4 , 16.07, -21.48]]), 'intercept*': array([4.]), 'accuracy': 0.9166666666666666} Weighted Vector :::: 400 {'Weight': array([[16. , -5.6 , 22.38, -25.54]]), 'intercept_': array([4.]), 'accuracy': 0.95}

++++++++++++++++++++ Kernel SVM ++++++++++++++++++

==accuracy 0.1 5 0.875 Weighted Vector :::: 200 {'Weight': 0, 'intercept'*: array([-0.23]), 'accuracy': 0.9166666666666666} Weighted Vector :::: 300 {'Weight': 0, 'intercept*': array([-0.15]), 'accuracy': 0.825} Weighted Vector :::: 400 {'Weight': 0, 'intercept_': array([-0.06]), 'accuracy': 0.875}

Weighted Vector for Linear SVM for 200 - [0.26, -0.29, 0.31, -0.38], intercept 0.02

Weighted Vector for Linear SVM for 300 - [0.01, -0.01, 0.07, -0.24], intercept 1.05

Weighted Vector for Linear SVM for 400 -[] 0.03, -0.03, 0.15, -0.23], intercept 0.31

Weighted Vector for Perceptron for 200 - [0.26, -0.29, 0.31, -0.38]

Weighted Vector for Perceptron for 300 - [5.5 , -1.4 , 16.07, -21.48]

Weighted Vector for Perceptron for 400 -[16. , -5.6 , 22.38, -25.54]

Kernel Intecept for 200 - 0.23,

Kernel Intecept for 300 - 0.15,

Kernel Intecept for 2400 - 0.06,

For Data set [200,300,400]

Accuracy for Linear SVM average = 0.98 + 0.81 + 0.95

Accuracy for Perceptron average = 0.97 + 0.91 + 0.95

Accuracy for Kernel SVM = 0.91 + 0.82 + 0.875

From the experiment it can be seen that Perceptron accuracy > Linear SVM > Kernel SVM.

The 'C' Parameter fined tuned for better accuracy for SVM Linear alogrithm using experiment approach of running the svm linear for different 'C' until optimal accuracy is achieved.

'C Parameter' = 0.025

The 'C' and "Gamma" Parameter fined tuned for better accuracy for SVM Linear alogrithm using experiment approach of running the svm kernel for different 'C' and 'gamma' until optimal accuracy is achieved.

"Gamma Parameter 0.2 and "C" Paramter 1

```
In [ ]:
```

Perceptron Linear SVM, Kernel SVM Data Set [500] Output

C Parameter fine tuned vlaue for better accuracy 1

Weighted Vector :::: 0 500 {'Weight': array([[1.81, -1.79, 1.8 , -1.79]]), 'intercept_': array([-0.13]), 'accuracy': 1.0}

++++++++++++++++++++++++++++++++Percepron+++++++++++++++++++

Weighted Vector :::: 500 {'Weight': array([[80.9 , -63.1 , 72.17, -69.57]]), 'intercept_': array([1.]), 'accuracy': 0.99}

+++++++++++++++++++++ Kernel SVM +++++++++++++++++++

==accuracy 0.1 2 0.97 Weighted Vector :::: 500 {'Weight': 0, 'intercept_': array([0.]), 'accuracy': 0.97}

For Data set [500]

Accuracy for Linear SVM average = 1

Accuracy for Perceptron average = 0.99

Accuracy for Kernel SVM = 0.97

From the experiment it can be seen that Linear SVM > Perceptron accuracy > Kernel SVM.

The 'C' Parameter fined tuned for better accuracy for SVM Linear alogrithm using experiment approach of running the svm linear for different 'C' until optimal accuracy is achieved.

'C Parameter' = 1

The 'C' and "Gamma" Parameter fined tuned for better accuracy for SVM Linear alogrithm using experiment approach of running the svm kernel for different 'C' and 'gamma' until optimal accuracy is achieved.

"Gamma Parameter 0.1 and "C" Parameter 2

--------[Data Set [500 200 300 400]--

++++++++++++++++++++++++++SVM++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy 1 Weighted Vector :::: 0 500 {'Weight': array([[1.8 , -1.77, 1.85, -1.83]]), 'intercept'*: array([-0.31]), 'accuracy': 1.0} Weighted Vector :::: 1 200 {'Weight': array([[1.37, -1.33, 1.36, -1.34]]), 'intercept*': array([-0.36]), 'accuracy': 1.0} Weighted Vector :::: 2 300 {'Weight': array([[1.43, -1.35, 1.55, -1.54]]), 'intercept'*: array([0.09]), 'accuracy': 1.0} Weighted Vector :::: 3 400 {'Weight': array([[1.43, -1.48, 1.68, -1.66]]), 'intercept*': array([-0.38]), 'accuracy': 1.0}

Accuracy (1+1+1+1)/4

+++++++++++++++++++++++++++++++Percepron++++++++++++++++++

Weighted Vector :::: 500 {'Weight': array([[73.7 , -63.1 , 79.97, -71.63]]), 'intercept'*: array([-7.]), 'accuracy': 1.0} Weighted Vector :::: 200 {'Weight': array([[58.4 , -60. , 54.86, -94.68]]), 'intercept*': array([-6.]), 'accuracy': 0.925} Weighted Vector :::: 300 {'Weight': array([[44.4 , -50.2 , 48.28, -50.34]]), 'intercept'*: array([0.]), 'accuracy': 1.0} Weighted Vector :::: 400 {'Weight': array([[64.3 , -72.7 , 80.57, -79.81]]), 'intercept*': array([-3.]), 'accuracy': 1.0}

Accuracy ---> (1 + .92 + 1 + 1)/4

W Vector --->

500 DP - [73.7 , -63.1 , 79.97, -71.63

400 DP 58.4 , -60. , 54.86, -94.68

300 DP 44.4 , -50.2 , 48.28, -50.34

200 64.3 , -72.7 , 80.57, -79.81

+++++++++++++++++++++ Kernel SVM ++++++++++++++++++

==accuracy 0.1 1 1.0 Weighted Vector :::: 500 {'Weight': 0, 'intercept'*: array([0.01]), 'accuracy': 1.0} Weighted Vector :::: 200 {'Weight': 0, 'intercept*': array([-0.1]), 'accuracy': 1.0} Weighted Vector :::: 300 {'Weight': 0, 'intercept'*: array([0.11]), 'accuracy': 0.9666666666666667} Weighted Vector :::: 400 {'Weight': 0, 'intercept*': array([0.06]), 'accuracy': 1.0}

Gamma -0.1

C Parameter 1

Based on the experiment done for linear and non-linear pattern, with fine tuning gamma and C parameter. It is observer Linear SVM is perfromed better then Perceptron and Finally Kernel SVM.

In [1]:
```
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import
from matplotlib import cm
import seaborn as sns
from sklearn.model_selection import train_test_split
import pdb
from sklearn.metrics import accuracy_score
```

Logic to create Data Set

In [2]:
```
# Generate random numbers
#data = np.random.random(-5,5,size=MAX_PATTERN)
# list of random float between a range -5 and 5
pd.options.display.float_format = '{:.2f}'.format
np.set_printoptions(precision=2)


def data_set_create(max):

    x_data = []
    MAX_PATTERN= max
    PER_REGION_PATTERN = 25 * max / 100


    # Set a length of the list to 10
    no_pattern = 1

    region_dict = { 'R13':0,'R14':0,'R24':0,'R23':0}

    r13 = []
    r14 = []
    r23 = []
    r24 = []

    region = [r13,r14,r23,r24]

    while (True):

        # any random float between 50.50 to MAX_PATTERN.50
        # don't use round() if you need number as it is
        #x1 = round(np.random.uniform(-5, 5), 1)
        x1 = round(np.random.uniform(-5, 5), 1)
        #x1.append(x)
        x2 = round(np.random.uniform(-5, 5), 1)
        #x2.append(x)
        # C1 ={(x1,x2)t :x1 >x2}andC2 ={(x1,x2)t :x1 <x2}.

        x1Sq = x1 * x1
        x2Sq = x2 * x2

        if (x1 == x2)or ((x1 + x2 + 1) == 0):
            continue
```

```
class_ = { 'C1':'No', 'C2':'No', 'C3':'No','C4':'No','R13':'No','R14':'No',
      'R23':'No','R24':'No', 'NC1':'No','NC2':'No', 'NC-Class':0}

if x1 > x2:
    class_['C1'] = 'Yes'
    class_['Y'] = 1

if x1 < x2 :
    class_['C2'] = 'Yes'
    class_['Y'] = 0
#C3 ={(x1,x2)t :x1 +x2 +1>0}andC4 ={(x1,x2)t :x1 +x2 +1<0}.

res = x1 + x2 + 1

if (res > 0 ):
    class_['C3'] = 'Yes'
if (res < 0) :
    class_['C4'] = 'Yes'

# R13 ={X :X ∈C1 andX ∈C3}, R14 ={X :X ∈C1 andX ∈C4} R24 ={X :X ∈C2 andX ∈C4},
# R23 ={X :X ∈C2 andX ∈C3}
if class_['C1'] == 'Yes' and class_['C3'] == 'Yes':
    if region_dict['R13'] >= PER_REGION_PATTERN:
        continue
    region_dict['R13'] = region_dict['R13'] + 1
    class_['R13']= 'Yes'

if class_['C1'] == 'Yes' and class_['C4'] == 'Yes':
    if region_dict['R14'] >= PER_REGION_PATTERN:
        continue
    class_['R14'] = 'Yes'
    region_dict['R14'] = region_dict['R14'] + 1


if class_['C2'] == 'Yes' and class_['C4'] == 'Yes':
    if region_dict['R24'] >= PER_REGION_PATTERN:
        continue
    class_['R24'] = 'Yes'
    region_dict['R24'] = region_dict['R24'] + 1

if class_['C2'] == 'Yes' and class_['C3'] == 'Yes':
    if region_dict['R23'] >= PER_REGION_PATTERN:
            continue
        class_['R23'] = 'Yes'
        region_dict['R23'] = region_dict['R23'] + 1

    if class_['R13'] == 'Yes' or  class_['R24'] == 'Yes':
        class_['NC1'] = 'Yes'
        class_['NC-Class'] = 1
    if class_['R14'] == 'Yes' or  class_['R23'] == 'Yes':
        class_['NC2'] = 'Yes'
        class_['NC-Class'] = 0

    d = [x1,x2,
               x1Sq,x2Sq,
               class_['Y'],
               class_['C1'],
               class_['C2'],
               class_['C3'],
               class_['C4'],
               class_['R13'],
               class_['R14'],
               class_['R24'],
               class_['R23'],
               class_['NC1'],
               class_['NC2'],
               class_['NC-Class']]

    if class_['R13'] == 'Yes':
        r13.append(d)

    if class_['R14'] == 'Yes':
        r14.append(d)

    if class_['R24'] == 'Yes':
        r24.append(d)

    if class_['R23'] == 'Yes':
        r23.append(d)

    x_data.append(d)


    no_pattern = no_pattern + 1
    if no_pattern == MAX_PATTERN + 1:
            break

    #print(f'No of DPs in Region ----> {region} \n')
    df_tmp = pd.DataFrame(x_data, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C4',
'R13','R14','R24','R23','NC1','NC2','NC-Class'])
    df_tmp.head()
    return(region)
```

Logic for SVM Kernel

In [3]:
```
def SVM_Kernel_fit(x_data, y_data, test_data):

    #test_x = test_data.drop(['Y','C1','C2','C3','C4','R13','R14','R24','R23','NC1','NC2','NC-Clas
s'],  axis=1)

    test_x = test_data.drop(['X1Square', 'X2Square','Y','C1','C2','C3','C4','R13','R14','R24','R23',
'NC1','NC2','NC-Class'],  axis=1)
    x_data = x_data.drop(['X1Square', 'X2Square'],axis=1)

    #print(f'{x_data}')
    test_y = test_data['NC-Class']



    from sklearn.svm import SVC

    from sklearn import svm

    # fit the model
    #for name, penalty in (('unreg', 1), ('reg', 0.05)):

    #clf =svm.SVC(kernel='rbf', random_state=1, gamma=0.4, C=1)

    max_g = 0
    max_c = 0
    m =0

    #for g,c in [ (0.1,1), (0.1,2), (0.1,3), (0.1,4),(0.2,1), (0.2,2), (0.2,3), (0.2,4), (0.3,1), (0.
3,2), (0.3,3), (0.3,4),
    #(0.2,2), (0.2,1), (0.2,3), (0.3,4),(0.3,2), (0.3,1), (0.3,3), (0.3,4 ), (0.4,1), (0.3,3), (0.4,4
) , (0.7,1), (0.7,2), (0.7,3), (0.7,4),(0.8,1), (0.8,2), (0.8,3), (0.8,4)]:
    #for g,c in [ (0.2,2), (0.2,1), (0.2,3), (0.3,4),(0.3,2), (0.3,1), (0.3,3), (0.3,4 ), (0.4,1),
 (0.3,3), (0.4,4 ) ]:

    #for g,c in [ (0.7,1), (0.7,2), (0.7,3), (0.7,4),(0.8,1), (0.8,2), (0.8,3), (0.8,4), (1,1), (1,
2), (1,3), (1,4) ]:

    for g in [.1,.2,.3,.4,.5,.6,.7,.8,.9 ]:
        for c in [1,2,3,4,5,6,7,8,9]:

            clf =svm.SVC(kernel='rbf', random_state=1, gamma=g, C=c)
            clf.fit(x_data, y_data)
            y_pred = clf.predict(test_x)
            # get the separating hyperplane

            acc = accuracy_score(test_y, y_pred)
            if (m < acc):
                m = acc
                max_g = g
                max_c = c
                intercept_ = clf.intercept_


            formatted_float = clf
            #print(f'coef  ---> {formatted_float}')
            #print("intercept  ---->",clf.intercept_)

    print(f'========================================================accuracy {max_g} {max_c} {m}')

    return ( m, 0, intercept_)
```

Logic for Linear SVM

In [4]:
```
def SVM_Linear_fit(x_data, y_data, test_data):

    test_x = test_data.drop(['Y','C1','C2','C3','C4','R13','R14','R24','R23','NC1','NC2','NC-Class'],
axis=1)

    test_y = test_data['NC-Class']

    from sklearn.svm import SVC

    from sklearn import svm

    # fit the model
    m = 0
    p = 0
    for penalty in [ 1, .8, .6,.4, .2, .1 , 0.05, 0.025]:
        clf = svm.SVC(kernel='linear', C=penalty)
        clf.fit(x_data, y_data)
        y_pred = clf.predict(test_x)
        # get the separating hyperplane

        formatted_float = clf.coef_
        #print(f'coef  {formatted_float}')


        acc = accuracy_score(test_y, y_pred)
        if m < acc:
            m = acc
            coef_ = clf.coef_
            intercept_ = clf.intercept_
            p = penalty

    print('C Parameter fine tuned vlaue for better accuracy ', p)
    return ( m, coef_, intercept_ )
```

Logic for perceptron classifier

```
In [5]: import pdb
cluster_list_pattern={}
Error_Threshold_List=[]

pd.options.display.float_format = '{:.2f}'.format

cluster_leader_list= []

accuracy_list = []


def perceptron_fit(x_data, y_data, test_data):
    from sklearn.datasets import load_digits
    from sklearn.linear_model import Perceptron

    test_x = test_data.drop(['Y','C1','C2','C3','C4','R13','R14','R24','R23','NC1','NC2','NC-Class'],
axis=1)
    test_y = test_data['NC-Class']
    clf = Perceptron(tol=1e-3, random_state=0)
    clf.fit(x_data, y_data)
    y_pred = clf.predict(test_x)

    #print(f' coef ---> {clf.coef_ }')
    #print(f' intercept ---> {clf.intercept_ }')

    acc = accuracy_score(test_y, y_pred)
    return ( acc, clf.coef_, clf.intercept_ )
```

Logic for dividing the Dataset for training and testing for 200,300,400

```
In [6]: '''
        data = [ [0,-0.5,0, 0.25],
        [-1,-0.5, 1, 0.25],
        [0.5, 0, -0.25, 0 ],
        [0.5, 1, -0.25, -1] ]
        '''




        from sklearn.model_selection import train_test_split

        w = {}
        index = '0'




        weight_perceptron_vector_list = []
        weight_SVM_Linear_vector_list = []
        weight_SVM_Kernel_vector_list = []


        count = 1
        size = [200,300,400]
        for max in size:

        #for max in [500]:
            reg =  data_set_create(max)
            df_train_all_region = []
            df_test_all_region = []

            for r in reg:
                #print(f'{r}')
                d = pd.DataFrame(r, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C4','R13',
        'R14','R24','R23','NC1','NC2','NC-Class'])
                d = d.reset_index(drop=True)

                #print(f' {d.iloc[0:max*25/100].to_string()}' )

                val = max /100  * .20

                train,test = train_test_split(d, train_size = val)
        df_train_all_region.append(train.iloc[5:10])

        df_test_all_region.append(test)


    _x_train = pd.concat(df_train_all_region)
    _x_train.sample(frac=1)
    _x_train = _x_train.reset_index(drop=True)


    _x_test = pd.concat(df_test_all_region)
    _x_test = _x_test.reset_index(drop=True)

    df_tmp = pd.DataFrame(_x_train, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C
4','R13','R14','R24','R23','NC1','NC2','NC-Class'])

    #print(df_tmp.to_string())
    df = df_tmp.drop(['Y','C1','C2','C3','C4','R13','R14','R24','R23','NC1','NC2','NC-Class'],  axis=
1)
    df_test = pd.DataFrame(_x_test, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C
4','R13','R14','R24','R23','NC1','NC2','NC-Class'])

    df = df.reset_index(drop=True)
    #print(df.to_string())



    w = 4
    h = 4
    d = 70
    plt.figure(figsize=(10, 4), dpi=d)
    st = f'Size[{max}] Feature'
    plt.xlabel(st, fontsize=14, color='black')
    plt.ylabel("X1 & X2 Feature ",fontsize=14, color='red')
    plt.title('Perceptron/SVM-Linear/SVM-Kernel Classification',fontsize=20, color='red')

    plt.plot(df['X1'], '*',color='g',markersize=8)
    plt.plot(df['X2'], 'o',color='r',markersize=8)

    print('\n--------[ Data Set [200 300 400 ]---------------------------------------------\n')
    for classifier in ['svm','percept','kernel-svm']:
    #for classifier in ['kernel-svm']:
```

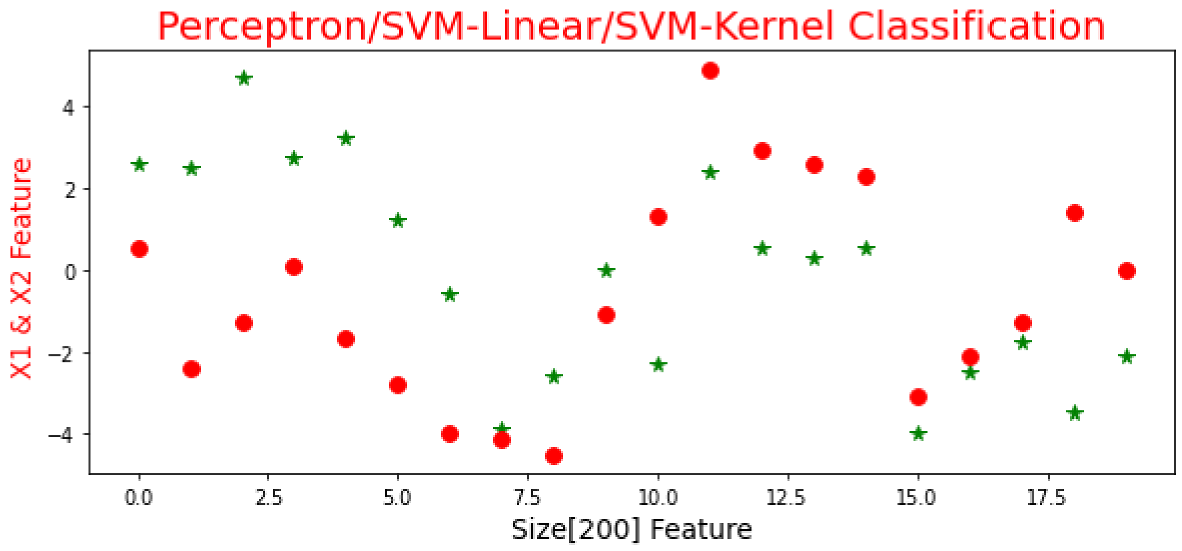
```python
print('\n-----------------------------------------------------\n')

if (classifier == 'percept'):

    print('\n-----------------------------------------------------\n')
    print('\n+++++++++++++++++++++++++++++++Perceptron++++++++++++++++++\n')
    (acc, coef_, intercept_ ) = perceptron_fit(df,df_tmp['NC-Class'],df_test)
    w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
    weight_perceptron_vector_list.append(w)

    acc_list =[]
    for i,w in enumerate(weight_perceptron_vector_list):
        acc_list.append(w['accuracy'])
        print( f'Weighted Vector :::: {size[i]} {w} ')
        #print( f'Weighted Vector :::: {i }  {w} ')
    if (count ==4):

        w = 4
        h = 4
        d = 70
        plt.figure(figsize=(10, 4), dpi=d)
        plt.xlabel(" size Feature ", fontsize=14, color='black')
        plt.ylabel("w['accuracy'] Feature ",fontsize=14, color='black')
        plt.title('Perceptron Classification',fontsize=20, color='black')
        plt.plot(size,acc_list,'o')

if (classifier == 'svm'):
    print('\n-----------------------------------------------------\n')
    print('\n+++++++++++++++++++++++++SVM++++++++++++++++++++++\n')

    (acc, coef_, intercept_ ) = SVM_Linear_fit(df,df_tmp['NC-Class'],df_test)



    w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
    weight_SVM_Linear_vector_list.append(w)

    acc_list =[]
    for i,w in enumerate(weight_SVM_Linear_vector_list):
        acc_list.append(w['accuracy'])
        print( f'Weighted Vector :::: {i } {size[i]} {w} ')
        #print( f'Weighted Vector :::: {i }  {w} ')
    if (count ==4):
            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
            plt.xlabel(" size Feature ", fontsize=14, color='black')
            plt.ylabel("w['accuracy'] Feature ",fontsize=20, color='black')
            plt.title('SVM  Linear Classification')
            plt.plot(size,acc_list,'o')

    if (classifier == 'kernel-svm'):
        print('\n------------------------------------------------------\n')
        print('\n+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++\n')


        from sklearn.svm import SVC

        from sklearn import svm

        #plot_decision_regions(df, df_tmp['NC-Class'], svm, test_idx=None, resolution=0.02)

        #weight_vector_list = []
        (acc, coef_, intercept_ ) = SVM_Kernel_fit(df,df_tmp['NC-Class'],df_test)
        w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
        weight_SVM_Kernel_vector_list.append(w)

        acc_list =[]
        for i,w in enumerate(weight_SVM_Kernel_vector_list):
            acc_list.append(w['accuracy'])
            print( f'Weighted Vector :::: {size[i]} {w} ')
            #print( f'Weighted Vector :::: {i }  {w} ')


        # Visualize the decision boundaries
        if (count ==4):

            d = { 'Size of Data ':size,'Accuracy':acc_list}
            df = pd.DataFrame(data=d)
            df.head(len(size))

            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
```

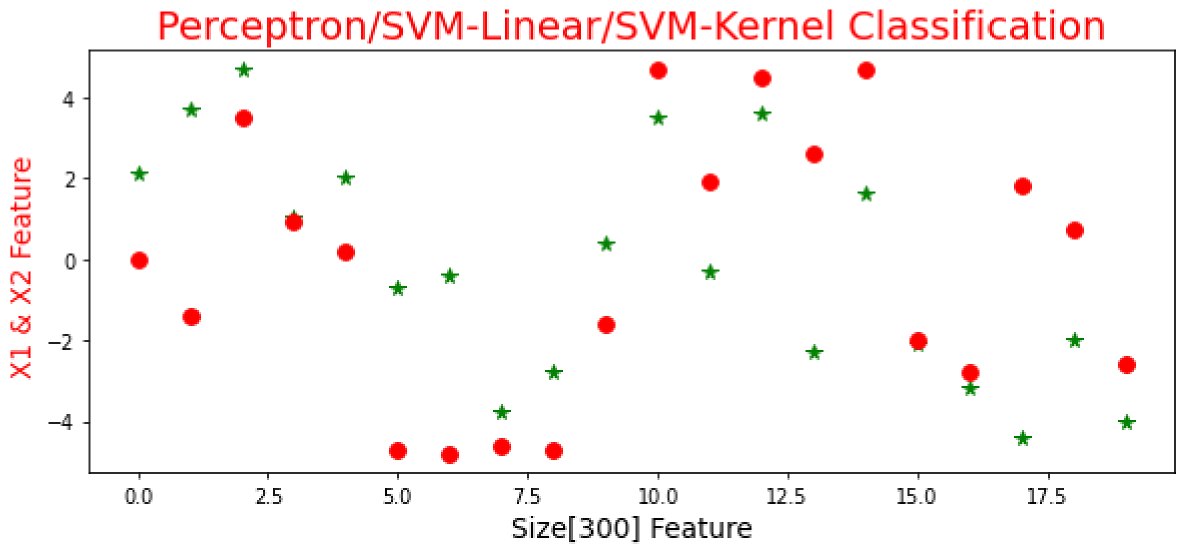
```
                plt.xlabel(" size Feature ", fontsize=12, color='black')
                plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
                plt.title('SVM KERNEL Classification')
                plt.plot(size,acc_list,'o')
    count = count + 1
```

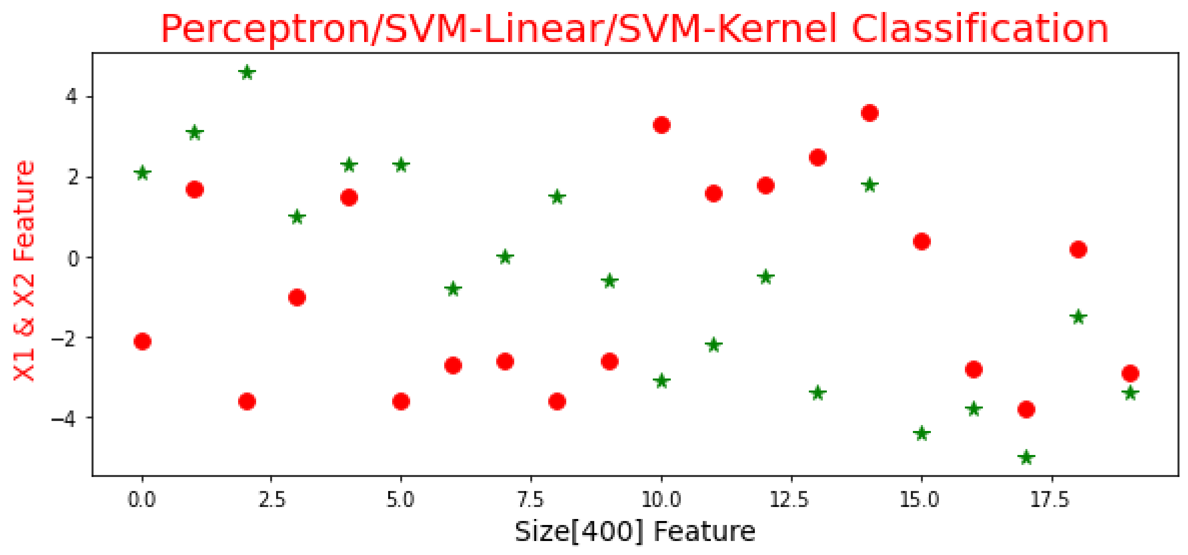
```
--------[ Data Set [200 300 400 ]---------------------------------------------


-----------------------------------------------------


-----------------------------------------------------


+++++++++++++++++++++++++++SVM+++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  0.1
Weighted Vector :::: 0 200 {'Weight': array([[ 0.17, -0.26,  0.25, -0.32]]), 'intercept_': array([0.
26]), 'accuracy': 0.9666666666666667}

-----------------------------------------------------


-----------------------------------------------------


++++++++++++++++++++++++++++++++Perceptron++++++++++++++++++

Weighted Vector :::: 200 {'Weight': array([[  3.9 , -10.8 ,  31.07, -23.42]]), 'intercept_': array
([4.]), 'accuracy': 0.9333333333333333}

-----------------------------------------------------


-----------------------------------------------------


+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++

============================================================accuracy 0.1 2 0.9166666666666666
Weighted Vector :::: 200 {'Weight': 0, 'intercept_': array([-0.04]), 'accuracy': 0.9166666666666666}

--------[ Data Set [200 300 400 ]---------------------------------------------


-----------------------------------------------------
```

```
-----------------------------------------------------


+++++++++++++++++++++++++++SVM+++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  0.1
Weighted Vector :::: 0 200 {'Weight': array([[ 0.17, -0.26,  0.25, -0.32]]), 'intercept_': array([0.
26]), 'accuracy': 0.9666666666666667}
Weighted Vector :::: 1 300 {'Weight': array([[ 0.01, -0.22,  0.23, -0.33]]), 'intercept_': array([0.
59]), 'accuracy': 0.9083333333333333}

-----------------------------------------------------


-----------------------------------------------------


++++++++++++++++++++++++++++++++Perceptron+++++++++++++++++++

Weighted Vector :::: 200 {'Weight': array([[  3.9 , -10.8 ,  31.07, -23.42]]), 'intercept_': array
([4.]), 'accuracy': 0.9333333333333333}
Weighted Vector :::: 300 {'Weight': array([[ -2.7 ,  -2.6 ,  22.79, -24.54]]), 'intercept_': array
([3.]), 'accuracy': 0.9333333333333333}

-----------------------------------------------------


-----------------------------------------------------


+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++

========================================================accuracy 0.3 2 0.85
Weighted Vector :::: 200 {'Weight': 0, 'intercept_': array([-0.04]), 'accuracy': 0.9166666666666666}
Weighted Vector :::: 300 {'Weight': 0, 'intercept_': array([0.02]), 'accuracy': 0.85}

--------[ Data Set [200 300 400 ]---------------------------------------------


-----------------------------------------------------


-----------------------------------------------------
```

```
++++++++++++++++++++++++++SVM++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  0.1
Weighted Vector :::: 0 200 {'Weight': array([[ 0.17, -0.26,  0.25, -0.32]]), 'intercept_': array([0.
26]), 'accuracy': 0.9666666666666667}
Weighted Vector :::: 1 300 {'Weight': array([[ 0.01, -0.22,  0.23, -0.33]]), 'intercept_': array([0.
59]), 'accuracy': 0.9083333333333333}
Weighted Vector :::: 2 400 {'Weight': array([[ 0.24, -0.36,  0.23, -0.38]]), 'intercept_': array([0.
63]), 'accuracy': 0.9625}

-----------------------------------------------------


-----------------------------------------------------


++++++++++++++++++++++++++++++++Perceptron++++++++++++++++++

Weighted Vector :::: 200 {'Weight': array([[  3.9 , -10.8 ,  31.07, -23.42]]), 'intercept_': array
([4.]), 'accuracy': 0.9333333333333333}
Weighted Vector :::: 300 {'Weight': array([[ -2.7 ,  -2.6 ,  22.79, -24.54]]), 'intercept_': array
([3.]), 'accuracy': 0.9333333333333333}
Weighted Vector :::: 400 {'Weight': array([[ 18.9 , -13.6 ,  22.41, -26.84]]), 'intercept_': array
([-3.]), 'accuracy': 0.9375}

-----------------------------------------------------


-----------------------------------------------------


+++++++++++++++++++++ Kernel SVM  ++++++++++++++++++

========================================================accuracy 0.1 2 0.9375
Weighted Vector :::: 200 {'Weight': 0, 'intercept_': array([-0.04]), 'accuracy': 0.9166666666666666}
Weighted Vector :::: 300 {'Weight': 0, 'intercept_': array([0.02]), 'accuracy': 0.85}
Weighted Vector :::: 400 {'Weight': 0, 'intercept_': array([0.12]), 'accuracy': 0.9375}
```


Perceptron/SVM-Linear/SVM-Kernel Classification
X1 & X2 Feature
Size[200] Feature
4
2
0
−2
−4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Perceptron/SVM-Linear/SVM-Kernel Classification
X1 & X2 Feature
Size[300] Feature
4
2
0
−2
−4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



Accuracy for SVM Kernel - size [200,300,400]

In [ ]:

```
```

In [7]:

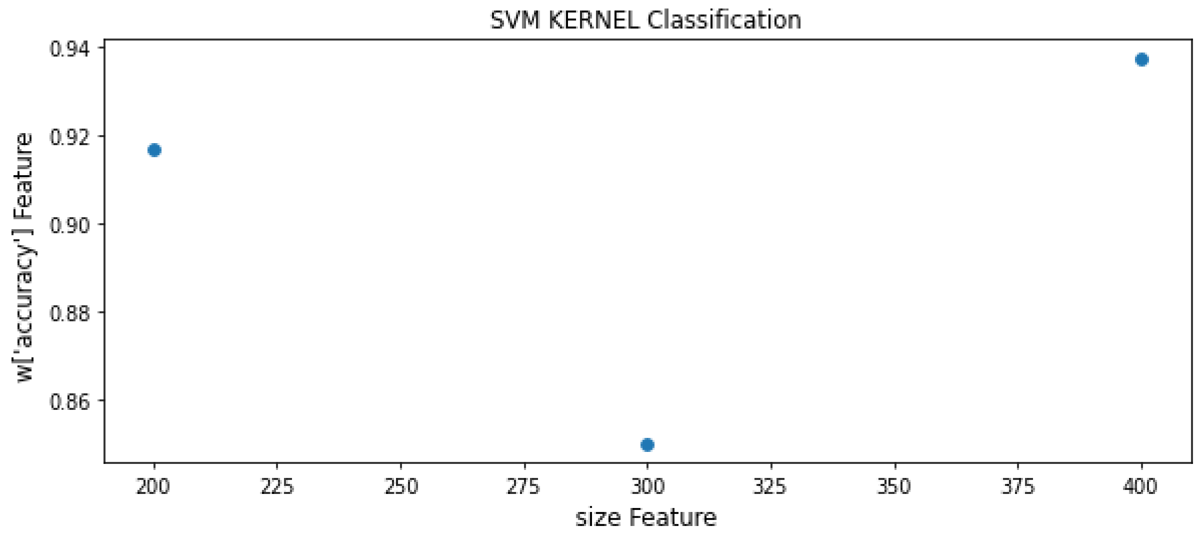
```
acc_list =[]
for i,w in enumerate(weight_SVM_Kernel_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for SVM Kernel ':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

Out[7]:

	Size of Data	Accuracy Calculation for SVM Kernel
0	200	0.92
1	300	0.85
2	400	0.94

```
In [8]: w = 4
        h = 4
        d = 70
        plt.figure(figsize=(10, 4), dpi=d)
        plt.xlabel(" size Feature ", fontsize=12, color='black')
        plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
        plt.title('SVM KERNEL Classification')
        plt.plot(size,acc_list,'o')
```

```
Out[8]: [<matplotlib.lines.Line2D at 0x125038490>]
```



Accuracy for SVM Linear - size [200,300,400]

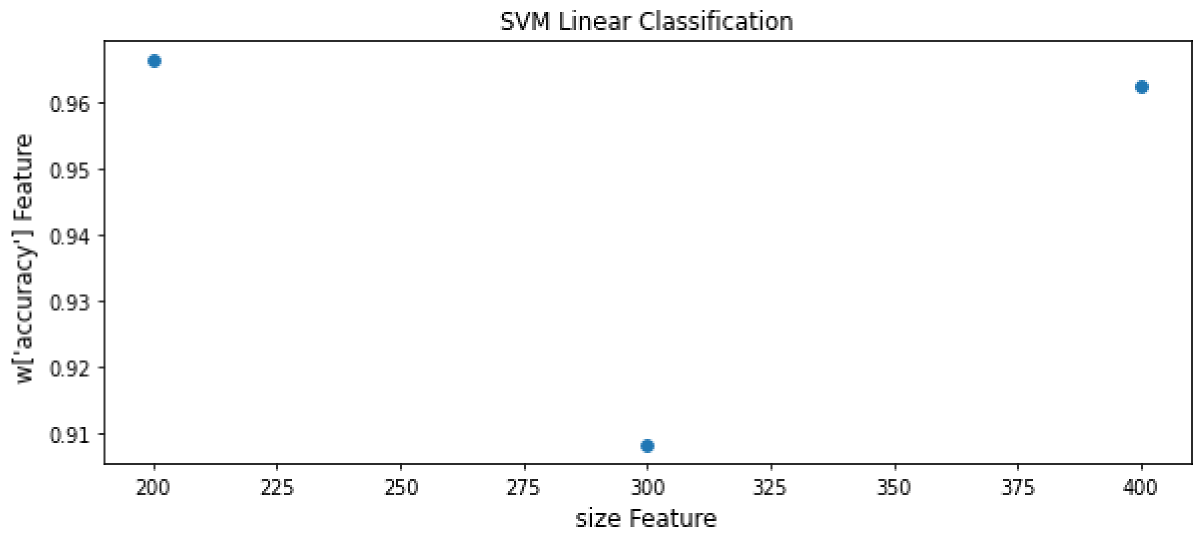
In [9]:
```
acc_list =[]
for i,w in enumerate(weight_SVM_Linear_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for SVM Linear ':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

Out[9]:

	Size of Data	Accuracy Calculation for SVM Linear
0	200	0.97
1	300	0.91
2	400	0.96

```
In [10]: w = 4
         h = 4
         d = 70
         plt.figure(figsize=(10, 4), dpi=d)
         plt.xlabel(" size Feature ", fontsize=12, color='black')
         plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
         plt.title('SVM Linear Classification')
         plt.plot(size,acc_list,'o')
```

```
Out[10]: [<matplotlib.lines.Line2D at 0x1250879a0>]
```



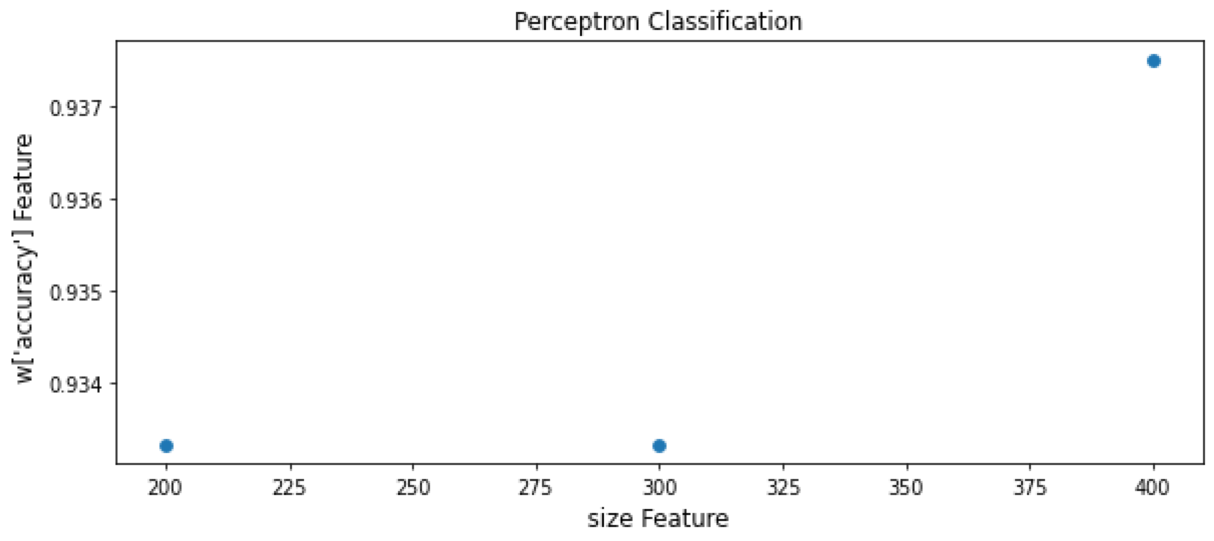
Accuracy for Perceptron - size [200,300,400]

In [11]:
```
acc_list =[]
for i,w in enumerate(weight_perceptron_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for Perceptron Algo ':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

Out[11]:

	Size of Data	Accuracy Calculation for Perceptron Algo
0	200	0.93
1	300	0.93
2	400	0.94

```
In [12]: w = 4
         h = 4
         d = 70
         plt.figure(figsize=(10, 4), dpi=d)
         plt.xlabel(" size Feature ", fontsize=12, color='black')
         plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
         plt.title('Perceptron Classification')
         plt.plot(size,acc_list,'o')

Out[12]: [<matplotlib.lines.Line2D at 0x1250e7550>]
```



Training Data set sample Output

```
In [13]: df_tmp.head(100)
```

Out[13]:

	X1	X2	X1Square	X2Square	Y	C1	C2	C3	C4	R13	R14	R24	R23	NC1	NC2	NC-Class
0	2.10	-2.10	4.41	4.41	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
1	3.10	1.70	9.61	2.89	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
2	4.60	-3.60	21.16	12.96	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
3	1.00	-1.00	1.00	1.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
4	2.30	1.50	5.29	2.25	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
5	2.30	-3.60	5.29	12.96	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
6	-0.80	-2.70	0.64	7.29	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
7	-0.00	-2.60	0.00	6.76	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
8	1.50	-3.60	2.25	12.96	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
9	-0.60	-2.60	0.36	6.76	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
10	-3.10	3.30	9.61	10.89	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
11	-2.20	1.60	4.84	2.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
12	-0.50	1.80	0.25	3.24	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
13	-3.40	2.50	11.56	6.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
14	1.80	3.60	3.24	12.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
15	-4.40	0.40	19.36	0.16	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
16	-3.80	-2.80	14.44	7.84	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
17	-5.00	-3.80	25.00	14.44	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
18	-1.50	0.20	2.25	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
19	-3.40	-2.90	11.56	8.41	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1

Testing Data set sample Output

```
In [14]: df_test.head(100)
```

Out[14]:

	X1	X2	X1Square	X2Square	Y	C1	C2	C3	C4	R13	R14	R24	R23	NC1	NC2	NC-Class
0	3.50	1.20	12.25	1.44	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
1	1.30	-1.00	1.69	1.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
2	2.90	0.70	8.41	0.49	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
3	4.10	-2.70	16.81	7.29	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
4	4.90	-4.40	24.01	19.36	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
75	-3.20	2.00	10.24	4.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
76	-4.20	1.90	17.64	3.61	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
77	-1.30	-0.20	1.69	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
78	-1.70	0.20	2.89	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
79	-3.30	-2.50	10.89	6.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1

80 rows × 16 columns

Running Perceptron/Linear/Kernel SVM for Dataset -500,200,300,400

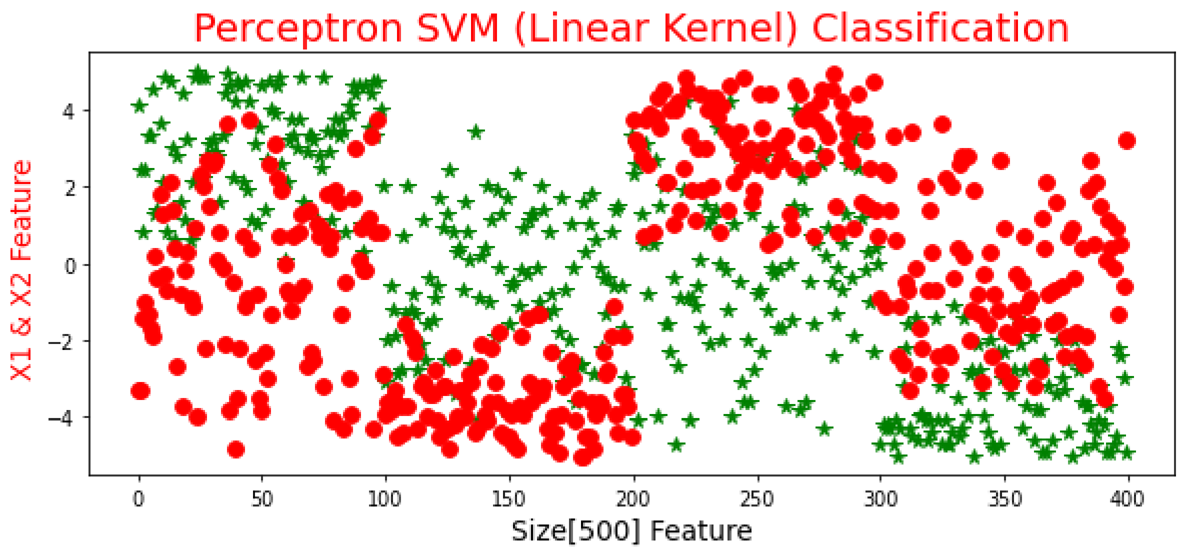
```
In [15]: from sklearn.model_selection import train_test_split

         w = {}
         index = '0'

         size = [500,200,300,400]

         weight_perceptron_vector_list = []
         weight_SVM_Linear_vector_list = []
         weight_SVM_Kernel_vector_list = []


         count = 1

         for max in size:
             reg =  data_set_create(max)
             df_train_all_region = []
             df_test_all_region = []

             for r in reg:
                 #print(f'{r}')
                 d = pd.DataFrame(r, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C4','R13',
         'R14','R24','R23','NC1','NC2','NC-Class'])
                 d = d.reset_index(drop=True)
                 #print(f' {d.iloc[0:max*25/100].to_string()}' )

                 train,test = train_test_split(d, train_size = 0.80)

                 df_train_all_region.append(train)
                 df_test_all_region.append(test)


             _x_train = pd.concat(df_train_all_region)
             _x_train = _x_train.reset_index(drop=True)

             _x_test = pd.concat(df_test_all_region)
             _x_test = _x_test.reset_index(drop=True)

             df_tmp = pd.DataFrame(_x_train, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C
         4','R13','R14','R24','R23','NC1','NC2','NC-Class'])

             #print(df_tmp.to_string())
```

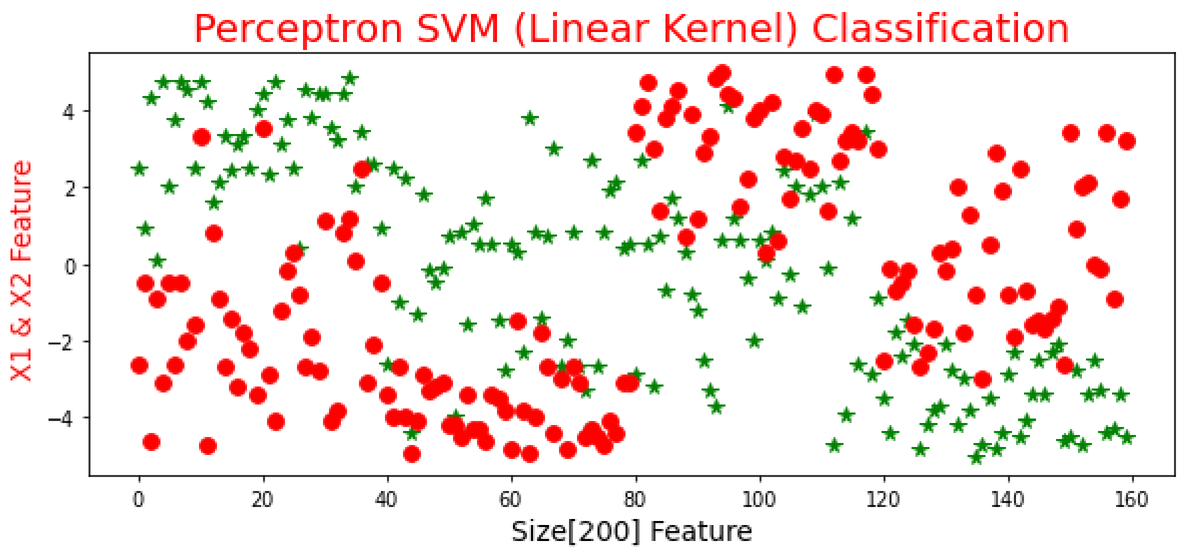
```
    df = df_tmp.drop(['Y','C1','C2','C3','C4','R13','R14','R24','R23','NC1','NC2','NC-Class'],  axis=
1)
    df_test = pd.DataFrame(_x_test, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C
4','R13','R14','R24','R23','NC1','NC2','NC-Class'])

    df = df.reset_index(drop=True)
    #print(df.to_string())



    w = 4
    h = 4
    d = 70
    plt.figure(figsize=(10, 4), dpi=d)
    st = f'Size[{max}] Feature'
    plt.xlabel(st, fontsize=14, color='black')
    plt.ylabel("X1 & X2 Feature ",fontsize=14, color='red')
    plt.title('Perceptron SVM (Linear Kernel) Classification',fontsize=20, color='red')

    plt.plot(df['X1'], '*',color='g',markersize=8)
    plt.plot(df['X2'], 'o',color='r',markersize=8)

    print('\n--------[ Data Set [ 500 200 300 400 ]---------------------------------------------\n')


    for classifier in ['svm','percept','kernel-svm']:
    #for classifier in ['kernel-svm']:



        if (classifier == 'percept'):

            print('\n-----------------------------------------------------\n')
            print('\n++++++++++++++++++++++++++++++++Percepron++++++++++++++++++\n')
            (acc, coef_, intercept_ ) = perceptron_fit(df,df_tmp['NC-Class'],df_test)
            w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
            weight_perceptron_vector_list.append(w)

            acc_list =[]
            for i,w in enumerate(weight_perceptron_vector_list):
                acc_list.append(w['accuracy'])
                print( f'Weighted Vector :::: {size[i]} {w} ')
                #print( f'Weighted Vector :::: {i }  {w} ')
        if (count ==4):

            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
            plt.xlabel(" size Feature ", fontsize=14, color='black')
            plt.ylabel("w['accuracy'] Feature ",fontsize=14, color='black')
            plt.title('Perceptron Classification',fontsize=20, color='black')
            plt.plot(size,acc_list,'o')

    if (classifier == 'svm'):
        print('\n-----------------------------------------------------\n')
        print('\n+++++++++++++++++++++++++++SVM++++++++++++++++++++++\n')

        (acc, coef_, intercept_ ) = SVM_Linear_fit(df,df_tmp['NC-Class'],df_test)



        w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
        weight_SVM_Linear_vector_list.append(w)

        acc_list =[]
        for i,w in enumerate(weight_SVM_Linear_vector_list):
            acc_list.append(w['accuracy'])
            print( f'Weighted Vector :::: {i } {size[i]} {w} ')
            #print( f'Weighted Vector :::: {i }  {w} ')
        if (count ==4):
            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
            plt.xlabel(" size Feature ", fontsize=14, color='black')
            plt.ylabel("w['accuracy'] Feature ",fontsize=20, color='black')
            plt.title('SVM  Linear Classification')
            plt.plot(size,acc_list,'o')

    if (classifier == 'kernel-svm'):
        print('\n-----------------------------------------------------\n')
        print('\n+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++\n')


        from sklearn.svm import SVC
        from sklearn import svm

        #plot_decision_regions(df, df_tmp['NC-Class'], svm, test_idx=None, resolution=0.02)

        #weight_vector_list = []
        (acc, coef_, intercept_ ) = SVM_Kernel_fit(df,df_tmp['NC-Class'],df_test)
        w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
        weight_SVM_Kernel_vector_list.append(w)

        acc_list =[]
        for i,w in enumerate(weight_SVM_Kernel_vector_list):
            acc_list.append(w['accuracy'])
            print( f'Weighted Vector :::: {size[i]} {w} ')
            #print( f'Weighted Vector :::: {i }  {w} ')


        # Visualize the decision boundaries
        if (count ==4):

            d = { 'Size of Data ':size,'Accuracy':acc_list}
            df = pd.DataFrame(data=d)
            df.head(len(size))

            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
            plt.xlabel(" size Feature ", fontsize=12, color='black')
            plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
            plt.title('SVM KERNEL Classification')
            plt.plot(size,acc_list,'o')
    count = count + 1
```

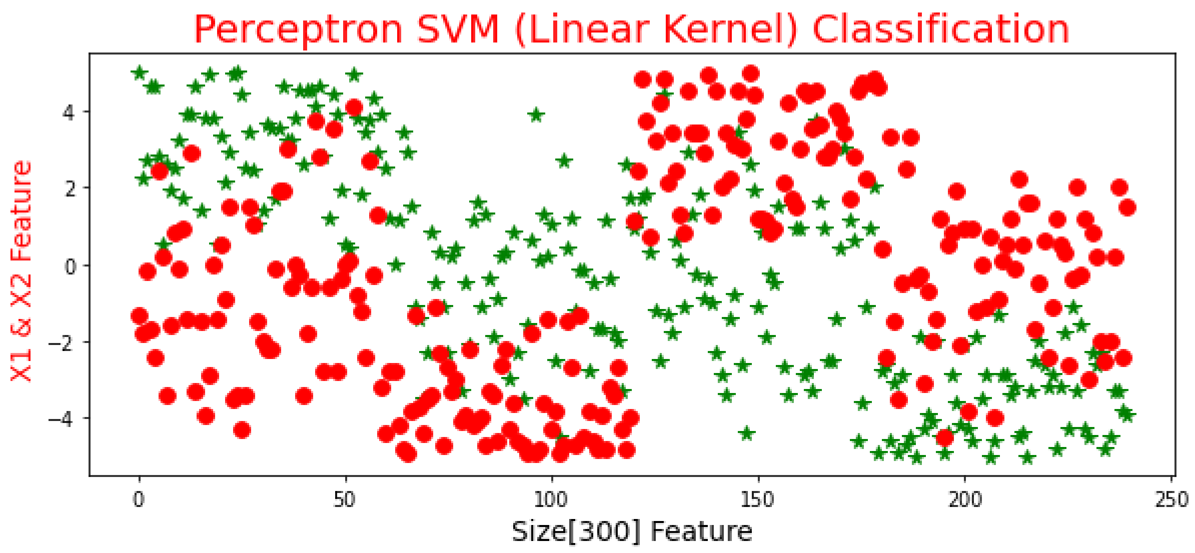
```
--------[ Data Set [ 500 200 300 400 ]---------------------------------------------


-----------------------------------------------------


++++++++++++++++++++++++++SVM++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  1
Weighted Vector :::: 0 500 {'Weight': array([[ 2.07, -1.86,  1.95, -1.98]]), 'intercept_': array([0.
12]), 'accuracy': 1.0}

-----------------------------------------------------


+++++++++++++++++++++++++++++++Percepron++++++++++++++++++

Weighted Vector :::: 500 {'Weight': array([[ 80.4 , -69.7 ,  85.84, -90.75]]), 'intercept_': array
([4.]), 'accuracy': 0.97}

-----------------------------------------------------


+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++

========================================================accuracy 0.1 1 0.99
Weighted Vector :::: 500 {'Weight': 0, 'intercept_': array([-0.03]), 'accuracy': 0.99}

--------[ Data Set [ 500 200 300 400 ]---------------------------------------------


-----------------------------------------------------


++++++++++++++++++++++++++SVM++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  1
Weighted Vector :::: 0 500 {'Weight': array([[ 2.07, -1.86,  1.95, -1.98]]), 'intercept_': array([0.
12]), 'accuracy': 1.0}
Weighted Vector :::: 1 200 {'Weight': array([[ 0.77, -0.88,  1.  , -1.  ]]), 'intercept_': array([-
0.23]), 'accuracy': 1.0}

-----------------------------------------------------
```

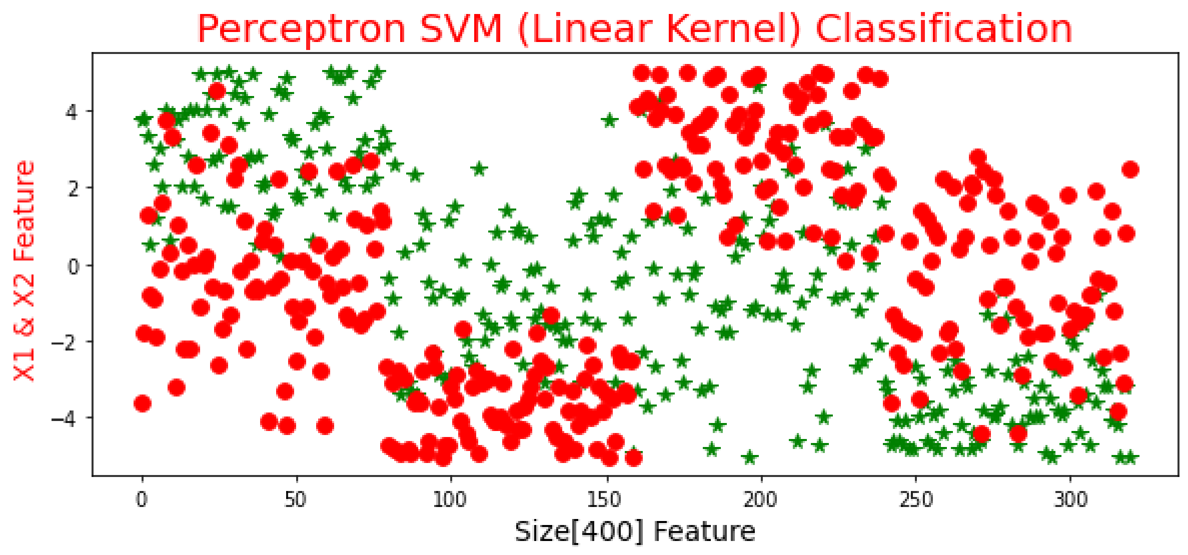
```
+++++++++++++++++++++++++++++++Percepron+++++++++++++++++

Weighted Vector :::: 500 {'Weight': array([[ 80.4 , -69.7 ,  85.84, -90.75]]), 'intercept_': array
([4.]), 'accuracy': 0.97}
Weighted Vector :::: 200 {'Weight': array([[ 60.6 , -47.9 ,  77.76, -79.11]]), 'intercept_': array
([8.]), 'accuracy': 0.975}

-----------------------------------------------------


++++++++++++++++++++ Kernel SVM  ++++++++++++++++++

========================================================accuracy 0.2 6 1.0
Weighted Vector :::: 500 {'Weight': 0, 'intercept_': array([-0.03]), 'accuracy': 0.99}
Weighted Vector :::: 200 {'Weight': 0, 'intercept_': array([-0.05]), 'accuracy': 1.0}

--------[ Data Set [ 500 200 300 400 ]-----------------------------------------------


-----------------------------------------------------


+++++++++++++++++++++++++++SVM+++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  1
Weighted Vector :::: 0 500 {'Weight': array([[ 2.07, -1.86,  1.95, -1.98]]), 'intercept_': array([0.
12]), 'accuracy': 1.0}
Weighted Vector :::: 1 200 {'Weight': array([[ 0.77, -0.88,  1.  , -1.  ]]), 'intercept_': array([-
0.23]), 'accuracy': 1.0}
Weighted Vector :::: 2 300 {'Weight': array([[ 1.6 , -1.59,  1.62, -1.66]]), 'intercept_': array([-
0.03]), 'accuracy': 1.0}

-----------------------------------------------------


+++++++++++++++++++++++++++++++Percepron+++++++++++++++++

Weighted Vector :::: 500 {'Weight': array([[ 80.4 , -69.7 ,  85.84, -90.75]]), 'intercept_': array
([4.]), 'accuracy': 0.97}
Weighted Vector :::: 200 {'Weight': array([[ 60.6 , -47.9 ,  77.76, -79.11]]), 'intercept_': array
([8.]), 'accuracy': 0.975}
```

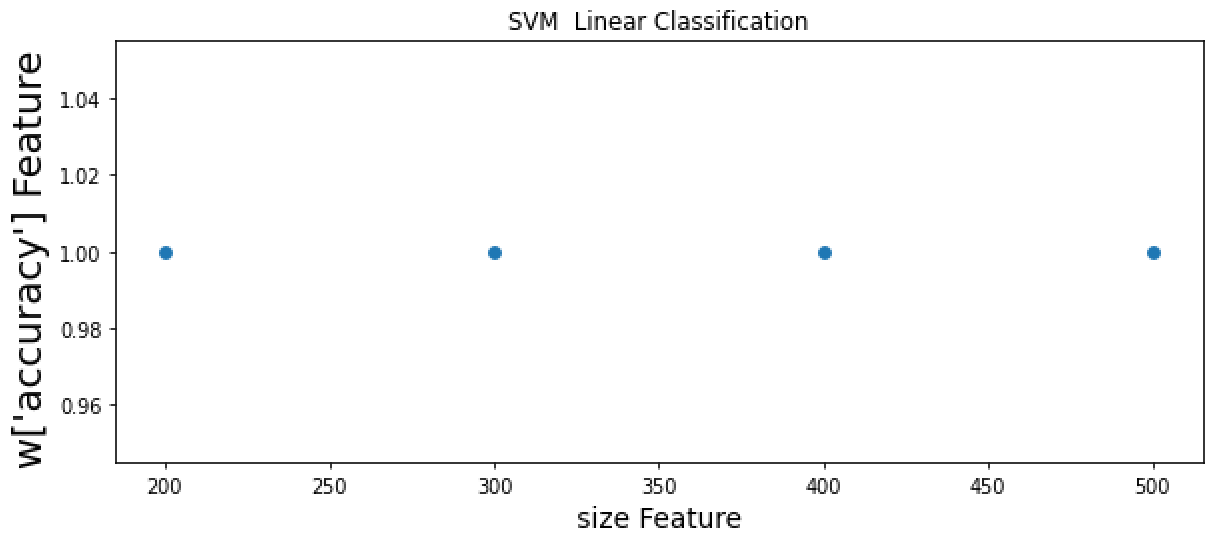
```
Weighted Vector :::: 300 {'Weight': array([[ 36.1 , -31.4 ,  36.01, -36.52]]), 'intercept_': array
([1.]), 'accuracy': 1.0}

-----------------------------------------------------


++++++++++++++++++++ Kernel SVM  ++++++++++++++++++

========================================================accuracy 0.5 2 1.0
Weighted Vector :::: 500 {'Weight': 0, 'intercept_': array([-0.03]), 'accuracy': 0.99}
Weighted Vector :::: 200 {'Weight': 0, 'intercept_': array([-0.05]), 'accuracy': 1.0}
Weighted Vector :::: 300 {'Weight': 0, 'intercept_': array([-0.01]), 'accuracy': 1.0}

--------[ Data Set [ 500 200 300 400 ]---------------------------------------------


-----------------------------------------------------


+++++++++++++++++++++++++++SVM+++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  0.4
Weighted Vector :::: 0 500 {'Weight': array([[ 2.07, -1.86,  1.95, -1.98]]), 'intercept_': array([0.
12]), 'accuracy': 1.0}
Weighted Vector :::: 1 200 {'Weight': array([[ 0.77, -0.88,  1.  , -1.  ]]), 'intercept_': array([-
0.23]), 'accuracy': 1.0}
Weighted Vector :::: 2 300 {'Weight': array([[ 1.6 , -1.59,  1.62, -1.66]]), 'intercept_': array([-
0.03]), 'accuracy': 1.0}
Weighted Vector :::: 3 400 {'Weight': array([[ 0.95, -1.18,  1.06, -1.13]]), 'intercept_': array([0.
18]), 'accuracy': 1.0}

-----------------------------------------------------


++++++++++++++++++++++++++++++++Percepron++++++++++++++++++

Weighted Vector :::: 500 {'Weight': array([[ 80.4 , -69.7 ,  85.84, -90.75]]), 'intercept_': array
([4.]), 'accuracy': 0.97}
Weighted Vector :::: 200 {'Weight': array([[ 60.6 , -47.9 ,  77.76, -79.11]]), 'intercept_': array
([8.]), 'accuracy': 0.975}
Weighted Vector :::: 300 {'Weight': array([[ 36.1 , -31.4 ,  36.01, -36.52]]), 'intercept_': array
([1.]), 'accuracy': 1.0}
Weighted Vector :::: 400 {'Weight': array([[ 43.3 , -50.  ,  53.45, -57.76]]), 'intercept_': array
([-1.]), 'accuracy': 1.0}

------------------------------------------------------


+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++

=========================================================accuracy 0.2 9 1.0
Weighted Vector :::: 500 {'Weight': 0, 'intercept_': array([-0.03]), 'accuracy': 0.99}
Weighted Vector :::: 200 {'Weight': 0, 'intercept_': array([-0.05]), 'accuracy': 1.0}
Weighted Vector :::: 300 {'Weight': 0, 'intercept_': array([-0.01]), 'accuracy': 1.0}
Weighted Vector :::: 400 {'Weight': 0, 'intercept_': array([-0.3]), 'accuracy': 1.0}
```

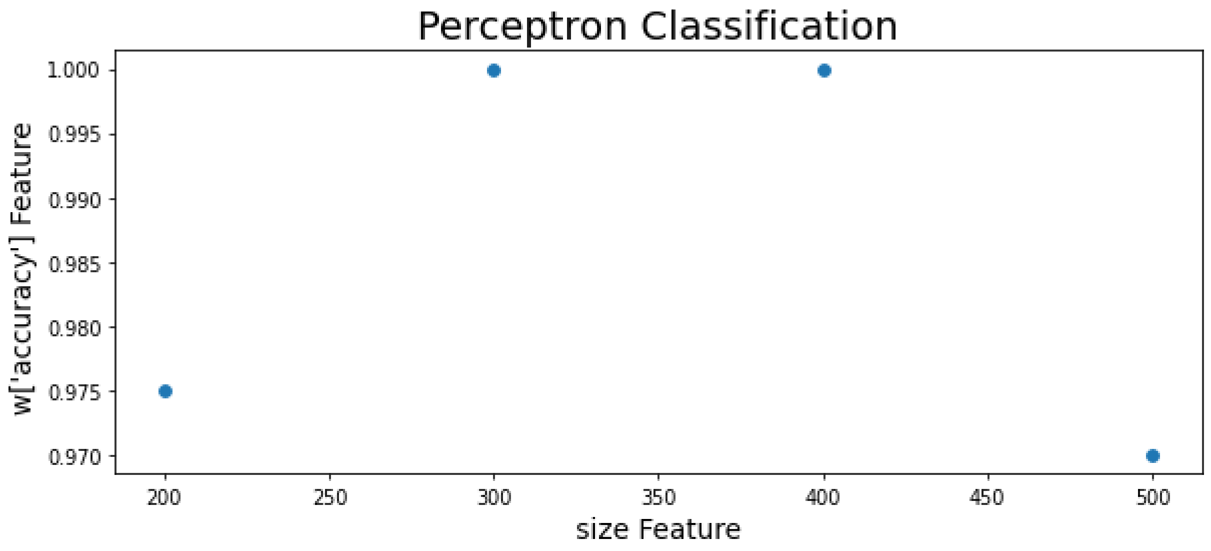


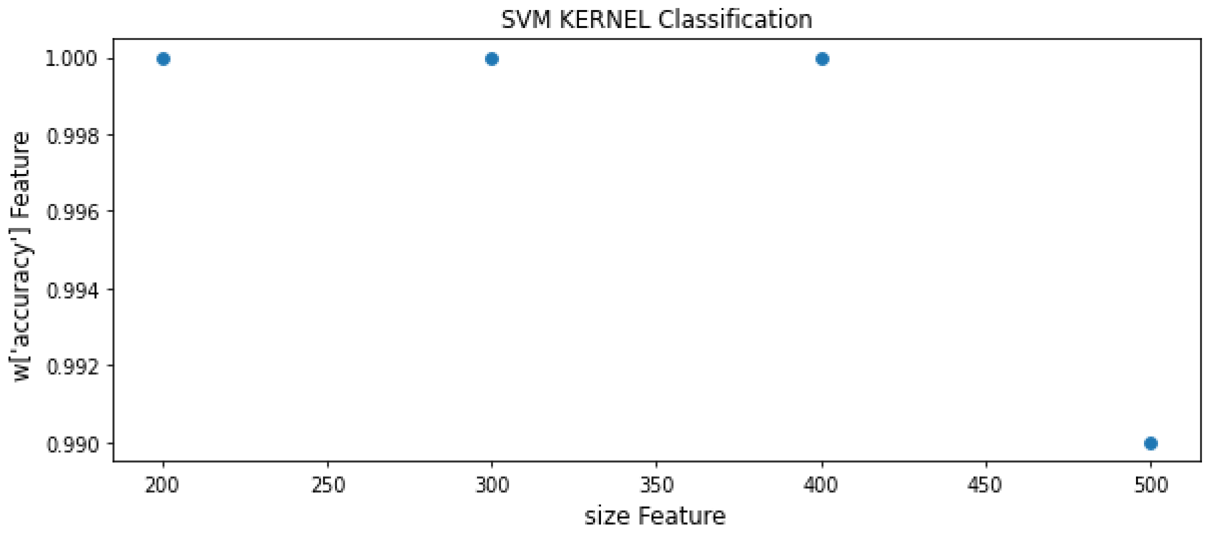

Perceptron SVM (Linear Kernel) Classification
X1 & X2 Feature
Size[200] Feature


Perceptron SVM (Linear Kernel) Classification
X1 & X2 Feature
Size[300] Feature


Perceptron SVM (Linear Kernel) Classification
X1 & X2 Feature
Size[400] Feature
4
2
0
−2
−4
0
50
100
150
200
250
300


SVM Linear Classification
w['accuracy'] Feature
size Feature
1.04
1.02
1.00
0.98
0.96
200
250
300
350
400
450
500





Accuracy for SVM Kernel - size [500,200,300,400]

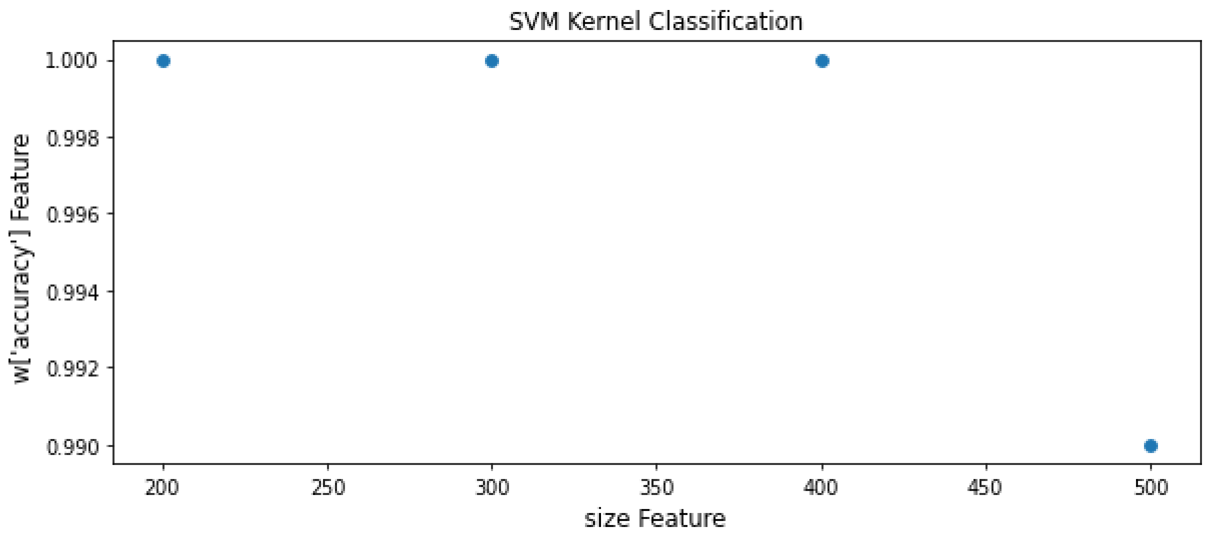
In [16]:
```
acc_list =[]
for i,w in enumerate(weight_SVM_Kernel_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for SVM Kernel ':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

Out[16]:

	Size of Data	Accuracy Calculation for SVM Kernel
0	500	0.99
1	200	1.00
2	300	1.00
3	400	1.00

In [17]:
```
w = 4
h = 4
d = 70
plt.figure(figsize=(10, 4), dpi=d)
plt.xlabel(" size Feature ", fontsize=12, color='black')
plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
plt.title('SVM Kernel Classification')
plt.plot(size,acc_list,'o')
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x125827a30>]
```



Accuracy for SVM Linear - size [500,200,300,400]

In [18]:
```
acc_list =[]
for i,w in enumerate(weight_SVM_Linear_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for SVM Linear ':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

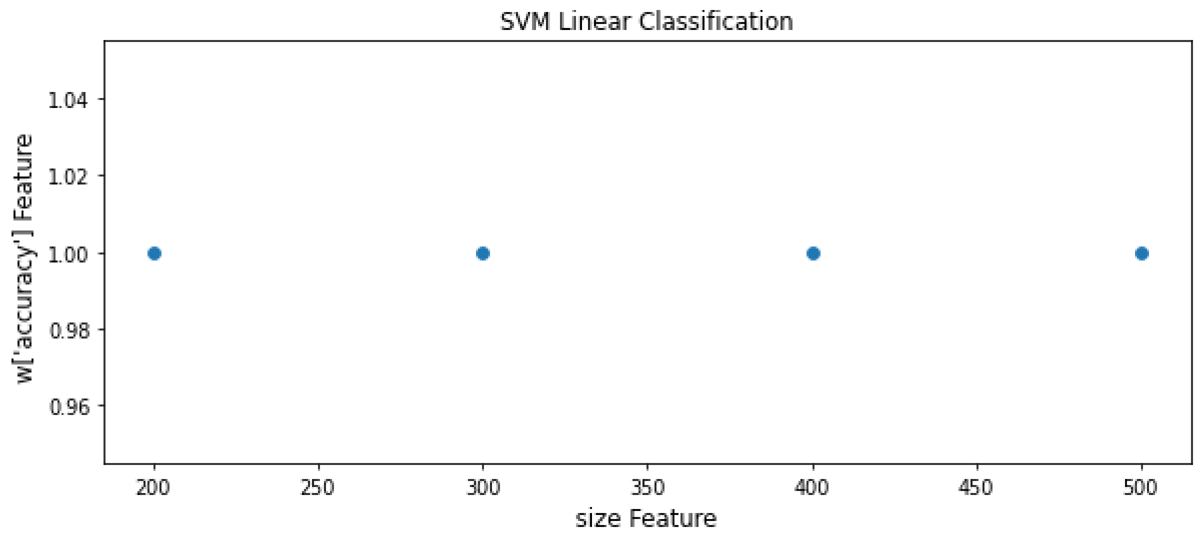
Out[18]:

	Size of Data	Accuracy Calculation for SVM Linear
0	500	1.00
1	200	1.00
2	300	1.00
3	400	1.00

In [ ]:

In [ ]:

```
In [19]: w = 4
         h = 4
         d = 70
         plt.figure(figsize=(10, 4), dpi=d)
         plt.xlabel(" size Feature ", fontsize=12, color='black')
         plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
         plt.title('SVM Linear Classification')
         plt.plot(size,acc_list,'o')

Out[19]: [<matplotlib.lines.Line2D at 0x12587fb50>]
```



Accuracy for Perceptron - size [500,200,300,400]

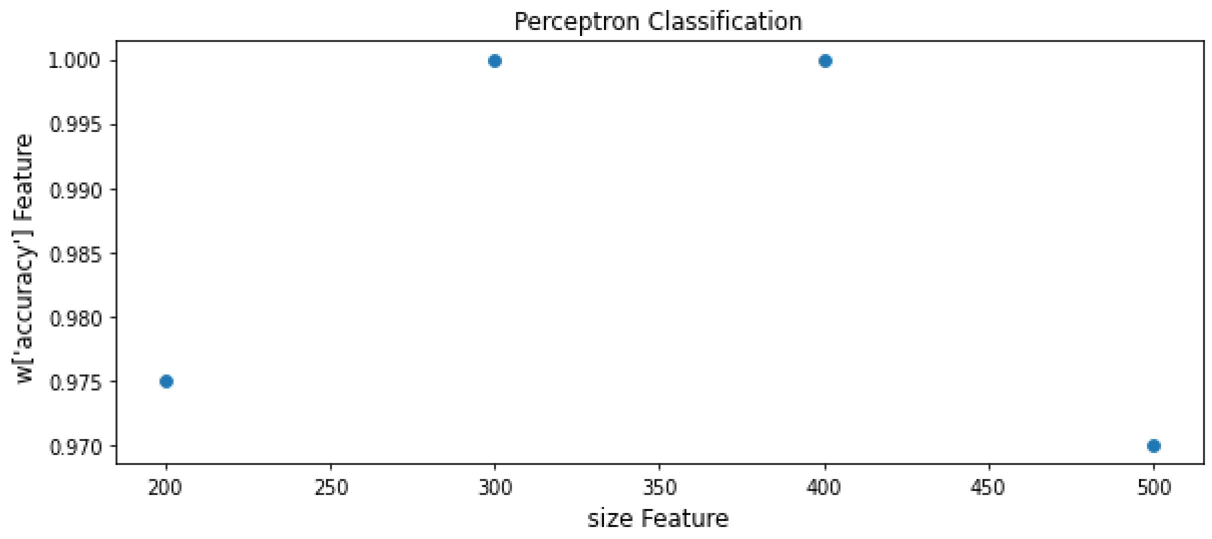
```
In [20]: acc_list =[]
         for i,w in enumerate(weight_perceptron_vector_list):
             acc_list.append(w['accuracy'])


         d = { 'Size of Data ':size,'Accuracy Calculation for Perceptron ':acc_list}
         df = pd.DataFrame(data=d)
         df.head(len(size))
```

Out[20]:

	Size of Data	Accuracy Calculation for Perceptron
0	500	0.97
1	200	0.97
2	300	1.00
3	400	1.00

```
In [ ]:
```

```
In [21]: w = 4
         h = 4
         d = 70
         plt.figure(figsize=(10, 4), dpi=d)
         plt.xlabel(" size Feature ", fontsize=12, color='black')
         plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
         plt.title('Perceptron Classification')
         plt.plot(size,acc_list,'o')

Out[21]: [<matplotlib.lines.Line2D at 0x1258d8730>]
```



Running Perceptron/Linear/Kernel SVM for Dataset -500

```
In [22]: from sklearn.model_selection import train_test_split

         w = {}
         index = '0'

         size = [500]

         weight_perceptron_vector_list = []
         weight_SVM_Linear_vector_list = []
         weight_SVM_Kernel_vector_list = []


         count = 1

         for max in size:
             reg =  data_set_create(max)
             df_train_all_region = []
             df_test_all_region = []

             for r in reg:
                 #print(f'{r}')
                 d = pd.DataFrame(r, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C4','R13',
         'R14','R24','R23','NC1','NC2','NC-Class'])
                 d = d.reset_index(drop=True)
                 #print(f' {d.iloc[0:max*25/100].to_string()}' )

                 train,test = train_test_split(d, train_size = 0.80)

                 df_train_all_region.append(train)
                 df_test_all_region.append(test)


             _x_train = pd.concat(df_train_all_region)
             _x_train = _x_train.reset_index(drop=True)

             _x_test = pd.concat(df_test_all_region)
             _x_test = _x_test.reset_index(drop=True)

             df_tmp = pd.DataFrame(_x_train, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C
         4','R13','R14','R24','R23','NC1','NC2','NC-Class'])

             print(df_tmp.to_string())
```

```
    df = df_tmp.drop(['Y','C1','C2','C3','C4','R13','R14','R24','R23','NC1','NC2','NC-Class'],  axis=
1)
    df_test = pd.DataFrame(_x_test, columns=['X1','X2', 'X1Square', 'X2Square','Y','C1','C2','C3','C
4','R13','R14','R24','R23','NC1','NC2','NC-Class'])

    df = df.reset_index(drop=True)
    print(df.to_string())



    w = 4
    h = 4
    d = 70
    plt.figure(figsize=(10, 4), dpi=d)
    st = f'Size[{max}] Feature'
    plt.xlabel(st, fontsize=14, color='black')
    plt.ylabel("X1 & X2 Feature ",fontsize=14, color='red')
    plt.title('Perceptron SVM (Linear Kernel) Classification',fontsize=20, color='red')

    plt.plot(df['X1'], '*',color='g',markersize=8)
    plt.plot(df['X2'], 'o',color='r',markersize=8)


    print('\n--------[ Data Set [500]---------------------------------------------\n')

    for classifier in ['svm','percept','kernel-svm']:
    #for classifier in ['kernel-svm']:



        if (classifier == 'percept'):

            print('\n-----------------------------------------------------\n')
            print('\n++++++++++++++++++++++++++++++++Percepron+++++++++++++++++\n')
            (acc, coef_, intercept_ ) = perceptron_fit(df,df_tmp['NC-Class'],df_test)
            w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
            weight_perceptron_vector_list.append(w)

            acc_list =[]
            for i,w in enumerate(weight_perceptron_vector_list):
                acc_list.append(w['accuracy'])
                print( f'Weighted Vector :::: {size[i]} {w} ')
                #print( f'Weighted Vector :::: {i }  {w} ')
        if (count ==4):

            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
            plt.xlabel(" size Feature ", fontsize=14, color='black')
            plt.ylabel("w['accuracy'] Feature ",fontsize=14, color='black')
            plt.title('Perceptron Classification',fontsize=20, color='black')
            plt.plot(size,acc_list,'o')

    if (classifier == 'svm'):
        print('\n-----------------------------------------------------\n')
        print('\n+++++++++++++++++++++++++++SVM+++++++++++++++++++++++\n')

        (acc, coef_, intercept_ ) = SVM_Linear_fit(df,df_tmp['NC-Class'],df_test)



        w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
        weight_SVM_Linear_vector_list.append(w)

        acc_list =[]
        for i,w in enumerate(weight_SVM_Linear_vector_list):
            acc_list.append(w['accuracy'])
            print( f'Weighted Vector :::: {i } {size[i]} {w} ')
            #print( f'Weighted Vector :::: {i }  {w} ')
        if (count ==4):
            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
            plt.xlabel(" size Feature ", fontsize=14, color='black')
            plt.ylabel("w['accuracy'] Feature ",fontsize=20, color='black')
            plt.title('SVM  Linear Classification')
            plt.plot(size,acc_list,'o')

    if (classifier == 'kernel-svm'):
        print('\n-----------------------------------------------------\n')
        print('\n+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++\n')


        from sklearn.svm import SVC
        from sklearn import svm

        #plot_decision_regions(df, df_tmp['NC-Class'], svm, test_idx=None, resolution=0.02)

        #weight_vector_list = []
        (acc, coef_, intercept_ ) = SVM_Kernel_fit(df,df_tmp['NC-Class'],df_test)
        w = { 'Weight'  : coef_, 'intercept_': intercept_ , 'accuracy': acc}
        weight_SVM_Kernel_vector_list.append(w)

        acc_list =[]
        for i,w in enumerate(weight_SVM_Kernel_vector_list):
            acc_list.append(w['accuracy'])
            print( f'Weighted Vector :::: {size[i]} {w} ')
            #print( f'Weighted Vector :::: {i }  {w} ')


        # Visualize the decision boundaries
        if (count ==4):

            d = { 'Size of Data ':size,'Accuracy':acc_list}
            df = pd.DataFrame(data=d)
            df.head(len(size))

            w = 4
            h = 4
            d = 70
            plt.figure(figsize=(10, 4), dpi=d)
            plt.xlabel(" size Feature ", fontsize=12, color='black')
            plt.ylabel("w['accuracy'] Feature ",fontsize=12, color='black')
            plt.title('SVM KERNEL Classification')
            plt.plot(size,acc_list,'o')
count = count + 1
```

```
      X1    X2  X1Square  X2Square  Y   C1   C2   C3   C4  R13  R14  R24  R23  NC1  NC2  NC-Class
0   4.90 -2.50     24.01      6.25  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
1   4.20 -4.40     17.64     19.36  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
2   1.80 -2.70      3.24      7.29  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
3   4.00 -3.60     16.00     12.96  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
4   4.50 -0.70     20.25      0.49  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
5   4.20  0.10     17.64      0.01  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
6   2.10 -2.70      4.41      7.29  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
7   4.10  3.20     16.81     10.24  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
8   0.90  0.30      0.81      0.09  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
9   1.90 -1.40      3.61      1.96  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
10  0.30 -0.90      0.09      0.81  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
11  2.70 -3.60      7.29     12.96  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
12  2.80  0.10      7.84      0.01  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
13  3.40 -3.20     11.56     10.24  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
14  4.40  0.20     19.36      0.04  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
15  2.70  1.30      7.29      1.69  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
16  0.70  0.30      0.49      0.09  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
17  4.70  4.60     22.09     21.16  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
18  3.90  3.60     15.21     12.96  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
19  4.20 -1.90     17.64      3.61  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
20  1.30  0.10      1.69      0.01  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
21  4.10 -2.80     16.81      7.84  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
22  4.70  4.40     22.09     19.36  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
23  2.70 -3.30      7.29     10.89  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
24  1.50  0.30      2.25      0.09  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
25  4.70  3.20     22.09     10.24  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
26  1.30 -0.40      1.69      0.16  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
27  4.30 -4.60     18.49     21.16  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
28  0.90  0.00      0.81      0.00  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
29  3.50  0.90     12.25      0.81  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
30  0.50 -0.20      0.25      0.04  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
31  2.90  2.20      8.41      4.84  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
32  3.50 -1.60     12.25      2.56  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
33  3.00  1.50      9.00      2.25  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
34  4.10 -3.30     16.81     10.89  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
35  1.30  0.20      1.69      0.04  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
36  1.10  0.70      1.21      0.49  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
37  3.20 -2.80     10.24      7.84  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
38  4.50  2.30     20.25      5.29  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
39  1.60  1.50      2.56      2.25  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
40  4.90 -0.30     24.01      0.09  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
41  3.50 -2.30     12.25      5.29  1  Yes   No  Yes   No  Yes   No   No   No  Yes   No         1
```

42	1.60	1.40	2.56	1.96	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
43	0.60	0.00	0.36	0.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
44	4.30	-0.80	18.49	0.64	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
45	3.60	-1.00	12.96	1.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
46	3.10	0.90	9.61	0.81	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
47	-0.20	-0.30	0.04	0.09	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
48	3.70	-4.10	13.69	16.81	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
49	2.20	0.20	4.84	0.04	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
50	4.90	-3.30	24.01	10.89	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
51	2.80	-2.50	7.84	6.25	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
52	4.50	-2.30	20.25	5.29	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
53	2.00	0.20	4.00	0.04	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
54	1.90	1.00	3.61	1.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
55	4.40	0.90	19.36	0.81	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
56	2.30	-2.00	5.29	4.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
57	3.60	1.20	12.96	1.44	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
58	3.10	0.60	9.61	0.36	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
59	-0.20	-0.50	0.04	0.25	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
60	5.00	2.30	25.00	5.29	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
61	4.40	3.40	19.36	11.56	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
62	2.40	-0.80	5.76	0.64	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
63	4.20	0.60	17.64	0.36	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
64	4.80	-3.10	23.04	9.61	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
65	4.80	3.10	23.04	9.61	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
66	2.80	2.70	7.84	7.29	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
67	4.40	-0.20	19.36	0.04	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
68	4.10	-3.20	16.81	10.24	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
69	2.90	1.90	8.41	3.61	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
70	2.80	-1.20	7.84	1.44	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
71	2.10	-2.00	4.41	4.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
72	0.40	0.10	0.16	0.01	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
73	2.50	0.50	6.25	0.25	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
74	1.50	0.10	2.25	0.01	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
75	2.10	-2.80	4.41	7.84	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
76	3.20	-0.30	10.24	0.09	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
77	2.80	-2.00	7.84	4.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
78	3.00	-2.50	9.00	6.25	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
79	2.70	-2.70	7.29	7.29	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
80	4.90	-0.70	24.01	0.49	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
81	4.70	-3.40	22.09	11.56	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
82	4.70	-1.70	22.09	2.89	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
83	3.50	1.90	12.25	3.61	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
84	4.60	1.40	21.16	1.96	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1

85	5.00	-2.60	25.00	6.76	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
86	1.50	1.30	2.25	1.69	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
87	3.40	-3.40	11.56	11.56	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
88	1.60	-1.80	2.56	3.24	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
89	4.40	1.80	19.36	3.24	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
90	4.90	0.30	24.01	0.09	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
91	3.30	-0.90	10.89	0.81	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
92	2.20	0.90	4.84	0.81	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
93	4.60	2.90	21.16	8.41	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
94	4.80	3.10	23.04	9.61	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
95	1.20	-0.60	1.44	0.36	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
96	3.90	2.00	15.21	4.00	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
97	0.70	-0.10	0.49	0.01	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
98	1.70	0.80	2.89	0.64	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
99	4.40	1.60	19.36	2.56	1	Yes	No	Yes	No	Yes	No	No	No	Yes	No	1
100	-3.70	-4.40	13.69	19.36	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
101	1.70	-4.70	2.89	22.09	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
102	0.10	-3.40	0.01	11.56	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
103	-0.20	-4.00	0.04	16.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
104	-2.20	-3.60	4.84	12.96	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
105	1.80	-3.30	3.24	10.89	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
106	0.50	-3.40	0.25	11.56	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
107	1.50	-3.40	2.25	11.56	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
108	1.00	-3.20	1.00	10.24	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
109	2.70	-4.70	7.29	22.09	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
110	-3.20	-3.40	10.24	11.56	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
111	-2.10	-3.50	4.41	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
112	-1.10	-3.70	1.21	13.69	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
113	0.30	-3.70	0.09	13.69	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
114	-2.70	-4.40	7.29	19.36	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
115	-1.30	-3.60	1.69	12.96	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
116	-1.90	-3.00	3.61	9.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
117	2.00	-3.50	4.00	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
118	-2.90	-3.20	8.41	10.24	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
119	-1.10	-3.30	1.21	10.89	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
120	0.20	-5.00	0.04	25.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
121	-0.10	-3.90	0.01	15.21	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
122	-2.70	-4.30	7.29	18.49	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
123	-0.20	-1.70	0.04	2.89	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
124	-1.90	-2.00	3.61	4.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
125	-0.20	-2.30	0.04	5.29	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
126	0.30	-1.40	0.09	1.96	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
127	-2.30	-3.30	5.29	10.89	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0

128	0.90	-4.10	0.81	16.81	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
129	2.00	-3.20	4.00	10.24	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
130	-0.90	-3.10	0.81	9.61	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
131	-1.40	-1.80	1.96	3.24	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
132	-2.60	-4.00	6.76	16.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
133	-4.20	-4.60	17.64	21.16	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
134	0.20	-3.70	0.04	13.69	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
135	-2.20	-3.50	4.84	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
136	-4.40	-4.90	19.36	24.01	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
137	-1.80	-2.80	3.24	7.84	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
138	-0.80	-0.90	0.64	0.81	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
139	3.40	-4.50	11.56	20.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
140	-1.30	-3.50	1.69	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
141	2.80	-4.20	7.84	17.64	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
142	0.10	-3.30	0.01	10.89	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
143	-2.00	-3.00	4.00	9.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
144	-2.60	-4.70	6.76	22.09	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
145	1.90	-4.90	3.61	24.01	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
146	-3.00	-3.90	9.00	15.21	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
147	-4.00	-4.90	16.00	24.01	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
148	-0.80	-2.90	0.64	8.41	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
149	0.50	-2.10	0.25	4.41	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
150	-4.10	-5.00	16.81	25.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
151	2.30	-3.90	5.29	15.21	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
152	-1.20	-2.00	1.44	4.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
153	0.50	-3.50	0.25	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
154	-1.50	-2.80	2.25	7.84	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
155	-0.10	-3.80	0.01	14.44	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
156	0.90	-2.30	0.81	5.29	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
157	1.70	-4.20	2.89	17.64	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
158	-0.30	-3.50	0.09	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
159	1.80	-4.60	3.24	21.16	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
160	1.10	-3.90	1.21	15.21	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
161	1.00	-3.00	1.00	9.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
162	-4.20	-4.40	17.64	19.36	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
163	0.90	-3.70	0.81	13.69	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
164	-1.70	-3.80	2.89	14.44	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
165	-1.00	-1.70	1.00	2.89	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
166	0.30	-3.90	0.09	15.21	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
167	-3.10	-4.60	9.61	21.16	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
168	-1.90	-2.60	3.61	6.76	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
169	0.80	-2.10	0.64	4.41	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
170	1.80	-4.20	3.24	17.64	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0

171	-4.80	-5.00	23.04	25.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
172	-1.30	-4.60	1.69	21.16	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
173	-1.10	-2.70	1.21	7.29	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
174	-0.60	-4.90	0.36	24.01	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
175	-1.50	-3.70	2.25	13.69	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
176	-3.30	-3.70	10.89	13.69	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
177	1.00	-4.40	1.00	19.36	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
178	1.00	-3.10	1.00	9.61	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
179	-1.00	-1.10	1.00	1.21	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
180	1.90	-4.80	3.61	23.04	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
181	-1.20	-4.70	1.44	22.09	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
182	1.80	-3.50	3.24	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
183	1.40	-3.50	1.96	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
184	-0.80	-2.70	0.64	7.29	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
185	-0.80	-4.40	0.64	19.36	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
186	0.20	-2.70	0.04	7.29	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
187	1.80	-3.90	3.24	15.21	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
188	-3.10	-4.30	9.61	18.49	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
189	-1.20	-3.20	1.44	10.24	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
190	-1.40	-1.90	1.96	3.61	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
191	-0.60	-2.00	0.36	4.00	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
192	-1.30	-1.50	1.69	2.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
193	-4.10	-4.50	16.81	20.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
194	-2.90	-3.20	8.41	10.24	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
195	1.10	-2.60	1.21	6.76	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
196	0.90	-2.60	0.81	6.76	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
197	-2.00	-2.20	4.00	4.84	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
198	-0.30	-3.30	0.09	10.89	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
199	-2.60	-3.50	6.76	12.25	1	Yes	No	No	Yes	No	Yes	No	No	No	Yes	0
200	-0.90	1.90	0.81	3.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
201	2.60	4.10	6.76	16.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
202	-1.40	0.70	1.96	0.49	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
203	-3.70	3.40	13.69	11.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
204	0.40	2.90	0.16	8.41	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
205	0.60	3.00	0.36	9.00	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
206	0.30	1.30	0.09	1.69	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
207	2.10	3.00	4.41	9.00	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
208	0.10	2.90	0.01	8.41	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
209	2.80	3.90	7.84	15.21	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
210	-2.80	4.50	7.84	20.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
211	2.10	3.60	4.41	12.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
212	-1.30	0.90	1.69	0.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
213	-1.30	0.90	1.69	0.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0

214	1.20	3.10	1.44	9.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
215	-0.70	3.40	0.49	11.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
216	-0.40	1.80	0.16	3.24	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
217	-0.90	3.60	0.81	12.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
218	2.20	3.20	4.84	10.24	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
219	-1.40	0.70	1.96	0.49	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
220	0.20	0.60	0.04	0.36	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
221	-1.20	2.90	1.44	8.41	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
222	-2.60	3.10	6.76	9.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
223	-1.50	1.60	2.25	2.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
224	-2.10	3.50	4.41	12.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
225	3.40	4.50	11.56	20.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
226	0.40	1.80	0.16	3.24	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
227	-1.50	1.60	2.25	2.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
228	-1.40	3.10	1.96	9.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
229	-0.50	1.90	0.25	3.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
230	-4.10	5.00	16.81	25.00	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
231	0.60	4.70	0.36	22.09	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
232	0.20	4.50	0.04	20.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
233	2.30	2.40	5.29	5.76	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
234	2.20	4.80	4.84	23.04	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
235	0.10	3.40	0.01	11.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
236	-0.40	2.50	0.16	6.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
237	1.80	2.70	3.24	7.29	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
238	-0.60	3.60	0.36	12.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
239	-4.50	4.30	20.25	18.49	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
240	-1.40	2.80	1.96	7.84	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
241	-3.40	3.90	11.56	15.21	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
242	2.90	4.50	8.41	20.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
243	-0.20	2.80	0.04	7.84	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
244	3.00	3.40	9.00	11.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
245	0.70	3.00	0.49	9.00	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
246	-3.70	3.60	13.69	12.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
247	-1.30	2.60	1.69	6.76	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
248	-2.00	2.20	4.00	4.84	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
249	-1.40	4.10	1.96	16.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
250	-0.10	4.40	0.01	19.36	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
251	-0.30	1.80	0.09	3.24	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
252	-2.50	4.10	6.25	16.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
253	-2.90	3.30	8.41	10.89	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
254	-0.90	0.10	0.81	0.01	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
255	-1.00	0.50	1.00	0.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
256	-2.60	1.90	6.76	3.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0

257	1.40	5.00	1.96	25.00	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
258	1.30	2.90	1.69	8.41	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
259	1.60	4.40	2.56	19.36	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
260	3.20	4.30	10.24	18.49	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
261	1.40	2.60	1.96	6.76	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
262	-1.30	2.40	1.69	5.76	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
263	1.90	3.60	3.61	12.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
264	-4.30	4.60	18.49	21.16	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
265	-3.60	2.90	12.96	8.41	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
266	-0.90	4.10	0.81	16.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
267	0.20	2.10	0.04	4.41	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
268	0.80	2.80	0.64	7.84	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
269	-1.80	1.40	3.24	1.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
270	-1.50	2.20	2.25	4.84	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
271	2.40	2.70	5.76	7.29	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
272	1.60	3.60	2.56	12.96	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
273	-1.50	1.20	2.25	1.44	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
274	1.70	3.80	2.89	14.44	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
275	1.20	3.90	1.44	15.21	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
276	-0.70	4.70	0.49	22.09	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
277	-0.20	-0.10	0.04	0.01	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
278	0.70	2.00	0.49	4.00	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
279	-0.40	1.90	0.16	3.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
280	0.50	0.60	0.25	0.36	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
281	-2.90	2.90	8.41	8.41	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
282	-3.10	3.20	9.61	10.24	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
283	-2.40	3.10	5.76	9.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
284	0.90	3.40	0.81	11.56	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
285	-4.10	4.50	16.81	20.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
286	1.10	1.50	1.21	2.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
287	-2.10	3.50	4.41	12.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
288	-4.50	4.10	20.25	16.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
289	1.00	2.70	1.00	7.29	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
290	2.90	3.70	8.41	13.69	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
291	0.70	3.90	0.49	15.21	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
292	2.10	4.10	4.41	16.81	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
293	-3.50	3.50	12.25	12.25	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
294	-2.40	2.40	5.76	5.76	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
295	-0.30	3.10	0.09	9.61	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
296	-0.50	2.80	0.25	7.84	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
297	-0.60	2.80	0.36	7.84	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
298	-0.80	4.80	0.64	23.04	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0
299	-0.40	2.40	0.16	5.76	0	No	Yes	Yes	No	No	No	No	Yes	No	Yes	0

300	-4.80	-3.60	23.04	12.96	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
301	-3.50	1.30	12.25	1.69	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
302	-3.80	1.00	14.44	1.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
303	-1.50	-1.10	2.25	1.21	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
304	-3.40	0.30	11.56	0.09	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
305	-3.10	-1.40	9.61	1.96	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
306	-4.00	0.50	16.00	0.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
307	-3.30	-2.00	10.89	4.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
308	-3.40	0.90	11.56	0.81	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
309	-3.50	0.70	12.25	0.49	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
310	-3.60	-3.20	12.96	10.24	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
311	-3.30	1.40	10.89	1.96	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
312	-2.40	-0.00	5.76	0.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
313	-1.70	-0.70	2.89	0.49	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
314	-2.50	-1.50	6.25	2.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
315	-2.90	-0.40	8.41	0.16	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
316	-4.30	0.30	18.49	0.09	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
317	-4.40	0.20	19.36	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
318	-3.70	-0.00	13.69	0.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
319	-4.90	-2.70	24.01	7.29	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
320	-2.40	1.20	5.76	1.44	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
321	-3.60	2.50	12.96	6.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
322	-4.10	0.20	16.81	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
323	-4.20	0.10	17.64	0.01	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
324	-1.80	-1.30	3.24	1.69	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
325	-4.50	-3.20	20.25	10.24	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
326	-3.40	-2.70	11.56	7.29	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
327	-2.80	0.90	7.84	0.81	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
328	-2.40	-1.20	5.76	1.44	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
329	-2.90	-0.10	8.41	0.01	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
330	-3.40	1.60	11.56	2.56	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
331	-1.90	0.50	3.61	0.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
332	-2.30	-0.00	5.29	0.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
333	-2.50	-1.90	6.25	3.61	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
334	-5.00	2.30	25.00	5.29	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
335	-3.00	-2.00	9.00	4.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
336	-2.90	0.10	8.41	0.01	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
337	-3.00	0.60	9.00	0.36	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
338	-4.20	-3.10	17.64	9.61	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
339	-1.70	0.10	2.89	0.01	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
340	-2.40	0.30	5.76	0.09	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
341	-2.70	-0.70	7.29	0.49	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
342	-4.00	2.50	16.00	6.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1

343	-1.40	-1.00	1.96	1.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
344	-1.70	-0.00	2.89	0.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
345	-3.20	-0.10	10.24	0.01	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
346	-4.50	-1.50	20.25	2.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
347	-4.90	-3.60	24.01	12.96	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
348	-4.40	-1.00	19.36	1.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
349	-3.50	-1.40	12.25	1.96	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
350	-1.30	-0.70	1.69	0.49	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
351	-4.50	3.40	20.25	11.56	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
352	-2.40	-0.10	5.76	0.01	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
353	-3.90	-2.50	15.21	6.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
354	-5.00	-4.60	25.00	21.16	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
355	-4.50	-0.20	20.25	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
356	-4.40	-1.60	19.36	2.56	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
357	-4.50	-2.60	20.25	6.76	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
358	-4.50	2.10	20.25	4.41	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
359	-1.10	-1.00	1.21	1.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
360	-4.30	-2.10	18.49	4.41	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
361	-2.10	0.60	4.41	0.36	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
362	-4.80	1.00	23.04	1.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
363	-3.60	2.10	12.96	4.41	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
364	-4.20	-3.80	17.64	14.44	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
365	-1.80	-1.20	3.24	1.44	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
366	-2.90	-1.50	8.41	2.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
367	-2.30	-2.00	5.29	4.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
368	-2.70	-0.50	7.29	0.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
369	-4.20	-0.60	17.64	0.36	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
370	-1.70	-0.80	2.89	0.64	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
371	-4.10	-2.30	16.81	5.29	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
372	-4.30	2.00	18.49	4.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
373	-4.10	-3.60	16.81	12.96	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
374	-4.60	-2.50	21.16	6.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
375	-2.30	0.20	5.29	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
376	-3.00	-2.90	9.00	8.41	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
377	-4.70	-3.20	22.09	10.24	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
378	-4.80	2.90	23.04	8.41	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
379	-3.40	0.60	11.56	0.36	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
380	-4.80	-2.60	23.04	6.76	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
381	-3.80	2.30	14.44	5.29	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
382	-1.60	-1.50	2.56	2.25	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
383	-4.30	1.00	18.49	1.00	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
384	-4.10	2.80	16.81	7.84	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1
385	-1.60	0.20	2.56	0.04	0	No	Yes	No	Yes	No	No	Yes	No	Yes	No	1

```
386 -3.40 -0.80     11.56      0.64  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
387 -4.60  1.00     21.16      1.00  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
388 -4.70  0.70     22.09      0.49  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
389 -1.40 -0.10      1.96      0.01  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
390 -3.40 -0.70     11.56      0.49  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
391 -4.80 -0.10     23.04      0.01  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
392 -3.40  0.70     11.56      0.49  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
393 -1.50 -1.30      2.25      1.69  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
394 -4.70 -3.50     22.09     12.25  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
395 -2.60  0.40      6.76      0.16  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
396 -4.80  0.20     23.04      0.04  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
397 -3.30  1.70     10.89      2.89  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
398 -4.70 -0.00     22.09      0.00  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
399 -4.70  3.00     22.09      9.00  0   No  Yes   No  Yes   No   No  Yes   No  Yes   No           1
       X1    X2  X1Square  X2Square
0    4.90 -2.50     24.01      6.25
1    4.20 -4.40     17.64     19.36
2    1.80 -2.70      3.24      7.29
3    4.00 -3.60     16.00     12.96
4    4.50 -0.70     20.25      0.49
5    4.20  0.10     17.64      0.01
6    2.10 -2.70      4.41      7.29
7    4.10  3.20     16.81     10.24
8    0.90  0.30      0.81      0.09
9    1.90 -1.40      3.61      1.96
10   0.30 -0.90      0.09      0.81
11   2.70 -3.60      7.29     12.96
12   2.80  0.10      7.84      0.01
13   3.40 -3.20     11.56     10.24
14   4.40  0.20     19.36      0.04
15   2.70  1.30      7.29      1.69
16   0.70  0.30      0.49      0.09
17   4.70  4.60     22.09     21.16
18   3.90  3.60     15.21     12.96
19   4.20 -1.90     17.64      3.61
20   1.30  0.10      1.69      0.01
21   4.10 -2.80     16.81      7.84
22   4.70  4.40     22.09     19.36
23   2.70 -3.30      7.29     10.89
24   1.50  0.30      2.25      0.09
25   4.70  3.20     22.09     10.24
26   1.30 -0.40      1.69      0.16
27   4.30 -4.60     18.49     21.16
```

```
28   0.90  0.00      0.81      0.00
29   3.50  0.90     12.25      0.81
30   0.50 -0.20      0.25      0.04
31   2.90  2.20      8.41      4.84
32   3.50 -1.60     12.25      2.56
33   3.00  1.50      9.00      2.25
34   4.10 -3.30     16.81     10.89
35   1.30  0.20      1.69      0.04
36   1.10  0.70      1.21      0.49
37   3.20 -2.80     10.24      7.84
38   4.50  2.30     20.25      5.29
39   1.60  1.50      2.56      2.25
40   4.90 -0.30     24.01      0.09
41   3.50 -2.30     12.25      5.29
42   1.60  1.40      2.56      1.96
43   0.60  0.00      0.36      0.00
44   4.30 -0.80     18.49      0.64
45   3.60 -1.00     12.96      1.00
46   3.10  0.90      9.61      0.81
47  -0.20 -0.30      0.04      0.09
48   3.70 -4.10     13.69     16.81
49   2.20  0.20      4.84      0.04
50   4.90 -3.30     24.01     10.89
51   2.80 -2.50      7.84      6.25
52   4.50 -2.30     20.25      5.29
53   2.00  0.20      4.00      0.04
54   1.90  1.00      3.61      1.00
55   4.40  0.90     19.36      0.81
56   2.30 -2.00      5.29      4.00
57   3.60  1.20     12.96      1.44
58   3.10  0.60      9.61      0.36
59  -0.20 -0.50      0.04      0.25
60   5.00  2.30     25.00      5.29
61   4.40  3.40     19.36     11.56
62   2.40 -0.80      5.76      0.64
63   4.20  0.60     17.64      0.36
64   4.80 -3.10     23.04      9.61
65   4.80  3.10     23.04      9.61
66   2.80  2.70      7.84      7.29
67   4.40 -0.20     19.36      0.04
68   4.10 -3.20     16.81     10.24
69   2.90  1.90      8.41      3.61
70   2.80 -1.20      7.84      1.44
```

```
71   2.10 -2.00      4.41      4.00
72   0.40  0.10      0.16      0.01
73   2.50  0.50      6.25      0.25
74   1.50  0.10      2.25      0.01
75   2.10 -2.80      4.41      7.84
76   3.20 -0.30     10.24      0.09
77   2.80 -2.00      7.84      4.00
78   3.00 -2.50      9.00      6.25
79   2.70 -2.70      7.29      7.29
80   4.90 -0.70     24.01      0.49
81   4.70 -3.40     22.09     11.56
82   4.70 -1.70     22.09      2.89
83   3.50  1.90     12.25      3.61
84   4.60  1.40     21.16      1.96
85   5.00 -2.60     25.00      6.76
86   1.50  1.30      2.25      1.69
87   3.40 -3.40     11.56     11.56
88   1.60 -1.80      2.56      3.24
89   4.40  1.80     19.36      3.24
90   4.90  0.30     24.01      0.09
91   3.30 -0.90     10.89      0.81
92   2.20  0.90      4.84      0.81
93   4.60  2.90     21.16      8.41
94   4.80  3.10     23.04      9.61
95   1.20 -0.60      1.44      0.36
96   3.90  2.00     15.21      4.00
97   0.70 -0.10      0.49      0.01
98   1.70  0.80      2.89      0.64
99   4.40  1.60     19.36      2.56
100 -3.70 -4.40     13.69     19.36
101  1.70 -4.70      2.89     22.09
102  0.10 -3.40      0.01     11.56
103 -0.20 -4.00      0.04     16.00
104 -2.20 -3.60      4.84     12.96
105  1.80 -3.30      3.24     10.89
106  0.50 -3.40      0.25     11.56
107  1.50 -3.40      2.25     11.56
108  1.00 -3.20      1.00     10.24
109  2.70 -4.70      7.29     22.09
110 -3.20 -3.40     10.24     11.56
111 -2.10 -3.50      4.41     12.25
112 -1.10 -3.70      1.21     13.69
113  0.30 -3.70      0.09     13.69
```

114	-2.70	-4.40	7.29	19.36
115	-1.30	-3.60	1.69	12.96
116	-1.90	-3.00	3.61	9.00
117	2.00	-3.50	4.00	12.25
118	-2.90	-3.20	8.41	10.24
119	-1.10	-3.30	1.21	10.89
120	0.20	-5.00	0.04	25.00
121	-0.10	-3.90	0.01	15.21
122	-2.70	-4.30	7.29	18.49
123	-0.20	-1.70	0.04	2.89
124	-1.90	-2.00	3.61	4.00
125	-0.20	-2.30	0.04	5.29
126	0.30	-1.40	0.09	1.96
127	-2.30	-3.30	5.29	10.89
128	0.90	-4.10	0.81	16.81
129	2.00	-3.20	4.00	10.24
130	-0.90	-3.10	0.81	9.61
131	-1.40	-1.80	1.96	3.24
132	-2.60	-4.00	6.76	16.00
133	-4.20	-4.60	17.64	21.16
134	0.20	-3.70	0.04	13.69
135	-2.20	-3.50	4.84	12.25
136	-4.40	-4.90	19.36	24.01
137	-1.80	-2.80	3.24	7.84
138	-0.80	-0.90	0.64	0.81
139	3.40	-4.50	11.56	20.25
140	-1.30	-3.50	1.69	12.25
141	2.80	-4.20	7.84	17.64
142	0.10	-3.30	0.01	10.89
143	-2.00	-3.00	4.00	9.00
144	-2.60	-4.70	6.76	22.09
145	1.90	-4.90	3.61	24.01
146	-3.00	-3.90	9.00	15.21
147	-4.00	-4.90	16.00	24.01
148	-0.80	-2.90	0.64	8.41
149	0.50	-2.10	0.25	4.41
150	-4.10	-5.00	16.81	25.00
151	2.30	-3.90	5.29	15.21
152	-1.20	-2.00	1.44	4.00
153	0.50	-3.50	0.25	12.25
154	-1.50	-2.80	2.25	7.84
155	-0.10	-3.80	0.01	14.44
156	0.90	-2.30	0.81	5.29

```
157  1.70 -4.20       2.89      17.64
158 -0.30 -3.50       0.09      12.25
159  1.80 -4.60       3.24      21.16
160  1.10 -3.90       1.21      15.21
161  1.00 -3.00       1.00       9.00
162 -4.20 -4.40      17.64      19.36
163  0.90 -3.70       0.81      13.69
164 -1.70 -3.80       2.89      14.44
165 -1.00 -1.70       1.00       2.89
166  0.30 -3.90       0.09      15.21
167 -3.10 -4.60       9.61      21.16
168 -1.90 -2.60       3.61       6.76
169  0.80 -2.10       0.64       4.41
170  1.80 -4.20       3.24      17.64
171 -4.80 -5.00      23.04      25.00
172 -1.30 -4.60       1.69      21.16
173 -1.10 -2.70       1.21       7.29
174 -0.60 -4.90       0.36      24.01
175 -1.50 -3.70       2.25      13.69
176 -3.30 -3.70      10.89      13.69
177  1.00 -4.40       1.00      19.36
178  1.00 -3.10       1.00       9.61
179 -1.00 -1.10       1.00       1.21
180  1.90 -4.80       3.61      23.04
181 -1.20 -4.70       1.44      22.09
182  1.80 -3.50       3.24      12.25
183  1.40 -3.50       1.96      12.25
184 -0.80 -2.70       0.64       7.29
185 -0.80 -4.40       0.64      19.36
186  0.20 -2.70       0.04       7.29
187  1.80 -3.90       3.24      15.21
188 -3.10 -4.30       9.61      18.49
189 -1.20 -3.20       1.44      10.24
190 -1.40 -1.90       1.96       3.61
191 -0.60 -2.00       0.36       4.00
192 -1.30 -1.50       1.69       2.25
193 -4.10 -4.50      16.81      20.25
194 -2.90 -3.20       8.41      10.24
195  1.10 -2.60       1.21       6.76
196  0.90 -2.60       0.81       6.76
197 -2.00 -2.20       4.00       4.84
198 -0.30 -3.30       0.09      10.89
199 -2.60 -3.50       6.76      12.25
```

```
200 -0.90  1.90      0.81      3.61
201  2.60  4.10      6.76     16.81
202 -1.40  0.70      1.96      0.49
203 -3.70  3.40     13.69     11.56
204  0.40  2.90      0.16      8.41
205  0.60  3.00      0.36      9.00
206  0.30  1.30      0.09      1.69
207  2.10  3.00      4.41      9.00
208  0.10  2.90      0.01      8.41
209  2.80  3.90      7.84     15.21
210 -2.80  4.50      7.84     20.25
211  2.10  3.60      4.41     12.96
212 -1.30  0.90      1.69      0.81
213 -1.30  0.90      1.69      0.81
214  1.20  3.10      1.44      9.61
215 -0.70  3.40      0.49     11.56
216 -0.40  1.80      0.16      3.24
217 -0.90  3.60      0.81     12.96
218  2.20  3.20      4.84     10.24
219 -1.40  0.70      1.96      0.49
220  0.20  0.60      0.04      0.36
221 -1.20  2.90      1.44      8.41
222 -2.60  3.10      6.76      9.61
223 -1.50  1.60      2.25      2.56
224 -2.10  3.50      4.41     12.25
225  3.40  4.50     11.56     20.25
226  0.40  1.80      0.16      3.24
227 -1.50  1.60      2.25      2.56
228 -1.40  3.10      1.96      9.61
229 -0.50  1.90      0.25      3.61
230 -4.10  5.00     16.81     25.00
231  0.60  4.70      0.36     22.09
232  0.20  4.50      0.04     20.25
233  2.30  2.40      5.29      5.76
234  2.20  4.80      4.84     23.04
235  0.10  3.40      0.01     11.56
236 -0.40  2.50      0.16      6.25
237  1.80  2.70      3.24      7.29
238 -0.60  3.60      0.36     12.96
239 -4.50  4.30     20.25     18.49
240 -1.40  2.80      1.96      7.84
241 -3.40  3.90     11.56     15.21
242  2.90  4.50      8.41     20.25
```

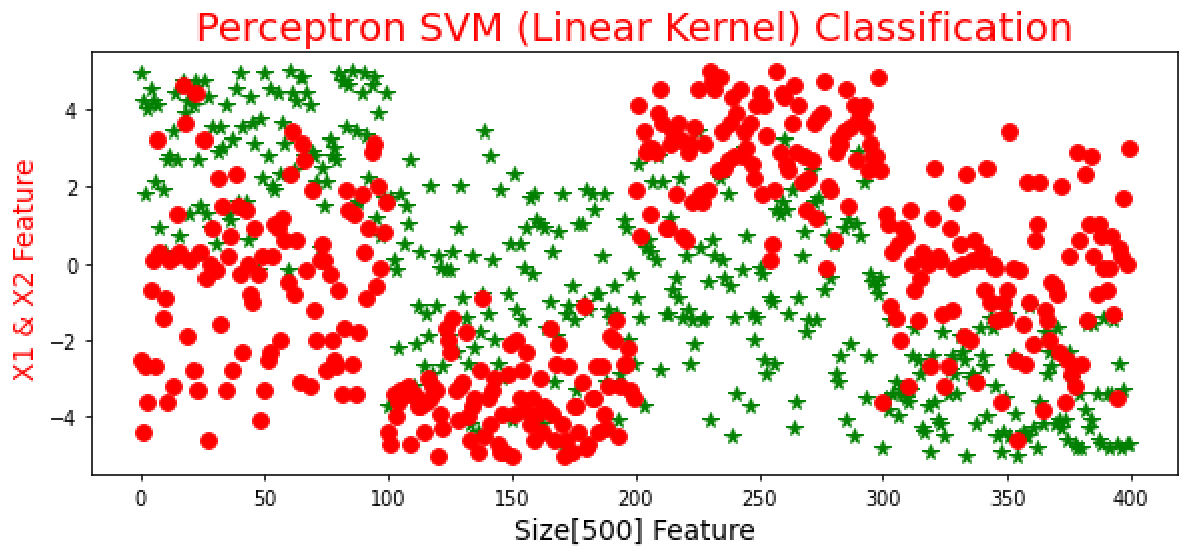
```
243 -0.20  2.80      0.04      7.84
244  3.00  3.40      9.00     11.56
245  0.70  3.00      0.49      9.00
246 -3.70  3.60     13.69     12.96
247 -1.30  2.60      1.69      6.76
248 -2.00  2.20      4.00      4.84
249 -1.40  4.10      1.96     16.81
250 -0.10  4.40      0.01     19.36
251 -0.30  1.80      0.09      3.24
252 -2.50  4.10      6.25     16.81
253 -2.90  3.30      8.41     10.89
254 -0.90  0.10      0.81      0.01
255 -1.00  0.50      1.00      0.25
256 -2.60  1.90      6.76      3.61
257  1.40  5.00      1.96     25.00
258  1.30  2.90      1.69      8.41
259  1.60  4.40      2.56     19.36
260  3.20  4.30     10.24     18.49
261  1.40  2.60      1.96      6.76
262 -1.30  2.40      1.69      5.76
263  1.90  3.60      3.61     12.96
264 -4.30  4.60     18.49     21.16
265 -3.60  2.90     12.96      8.41
266 -0.90  4.10      0.81     16.81
267  0.20  2.10      0.04      4.41
268  0.80  2.80      0.64      7.84
269 -1.80  1.40      3.24      1.96
270 -1.50  2.20      2.25      4.84
271  2.40  2.70      5.76      7.29
272  1.60  3.60      2.56     12.96
273 -1.50  1.20      2.25      1.44
274  1.70  3.80      2.89     14.44
275  1.20  3.90      1.44     15.21
276 -0.70  4.70      0.49     22.09
277 -0.20 -0.10      0.04      0.01
278  0.70  2.00      0.49      4.00
279 -0.40  1.90      0.16      3.61
280  0.50  0.60      0.25      0.36
281 -2.90  2.90      8.41      8.41
282 -3.10  3.20      9.61     10.24
283 -2.40  3.10      5.76      9.61
284  0.90  3.40      0.81     11.56
285 -4.10  4.50     16.81     20.25
```

```
286  1.10  1.50      1.21      2.25
287 -2.10  3.50      4.41     12.25
288 -4.50  4.10     20.25     16.81
289  1.00  2.70      1.00      7.29
290  2.90  3.70      8.41     13.69
291  0.70  3.90      0.49     15.21
292  2.10  4.10      4.41     16.81
293 -3.50  3.50     12.25     12.25
294 -2.40  2.40      5.76      5.76
295 -0.30  3.10      0.09      9.61
296 -0.50  2.80      0.25      7.84
297 -0.60  2.80      0.36      7.84
298 -0.80  4.80      0.64     23.04
299 -0.40  2.40      0.16      5.76
300 -4.80 -3.60     23.04     12.96
301 -3.50  1.30     12.25      1.69
302 -3.80  1.00     14.44      1.00
303 -1.50 -1.10      2.25      1.21
304 -3.40  0.30     11.56      0.09
305 -3.10 -1.40      9.61      1.96
306 -4.00  0.50     16.00      0.25
307 -3.30 -2.00     10.89      4.00
308 -3.40  0.90     11.56      0.81
309 -3.50  0.70     12.25      0.49
310 -3.60 -3.20     12.96     10.24
311 -3.30  1.40     10.89      1.96
312 -2.40 -0.00      5.76      0.00
313 -1.70 -0.70      2.89      0.49
314 -2.50 -1.50      6.25      2.25
315 -2.90 -0.40      8.41      0.16
316 -4.30  0.30     18.49      0.09
317 -4.40  0.20     19.36      0.04
318 -3.70 -0.00     13.69      0.00
319 -4.90 -2.70     24.01      7.29
320 -2.40  1.20      5.76      1.44
321 -3.60  2.50     12.96      6.25
322 -4.10  0.20     16.81      0.04
323 -4.20  0.10     17.64      0.01
324 -1.80 -1.30      3.24      1.69
325 -4.50 -3.20     20.25     10.24
326 -3.40 -2.70     11.56      7.29
327 -2.80  0.90      7.84      0.81
328 -2.40 -1.20      5.76      1.44
```

```
329 -2.90 -0.10      8.41      0.01
330 -3.40  1.60     11.56      2.56
331 -1.90  0.50      3.61      0.25
332 -2.30 -0.00      5.29      0.00
333 -2.50 -1.90      6.25      3.61
334 -5.00  2.30     25.00      5.29
335 -3.00 -2.00      9.00      4.00
336 -2.90  0.10      8.41      0.01
337 -3.00  0.60      9.00      0.36
338 -4.20 -3.10     17.64      9.61
339 -1.70  0.10      2.89      0.01
340 -2.40  0.30      5.76      0.09
341 -2.70 -0.70      7.29      0.49
342 -4.00  2.50     16.00      6.25
343 -1.40 -1.00      1.96      1.00
344 -1.70 -0.00      2.89      0.00
345 -3.20 -0.10     10.24      0.01
346 -4.50 -1.50     20.25      2.25
347 -4.90 -3.60     24.01     12.96
348 -4.40 -1.00     19.36      1.00
349 -3.50 -1.40     12.25      1.96
350 -1.30 -0.70      1.69      0.49
351 -4.50  3.40     20.25     11.56
352 -2.40 -0.10      5.76      0.01
353 -3.90 -2.50     15.21      6.25
354 -5.00 -4.60     25.00     21.16
355 -4.50 -0.20     20.25      0.04
356 -4.40 -1.60     19.36      2.56
357 -4.50 -2.60     20.25      6.76
358 -4.50  2.10     20.25      4.41
359 -1.10 -1.00      1.21      1.00
360 -4.30 -2.10     18.49      4.41
361 -2.10  0.60      4.41      0.36
362 -4.80  1.00     23.04      1.00
363 -3.60  2.10     12.96      4.41
364 -4.20 -3.80     17.64     14.44
365 -1.80 -1.20      3.24      1.44
366 -2.90 -1.50      8.41      2.25
367 -2.30 -2.00      5.29      4.00
368 -2.70 -0.50      7.29      0.25
369 -4.20 -0.60     17.64      0.36
370 -1.70 -0.80      2.89      0.64
371 -4.10 -2.30     16.81      5.29
```

```
372 -4.30  2.00     18.49      4.00
373 -4.10 -3.60     16.81     12.96
374 -4.60 -2.50     21.16      6.25
375 -2.30  0.20      5.29      0.04
376 -3.00 -2.90      9.00      8.41
377 -4.70 -3.20     22.09     10.24
378 -4.80  2.90     23.04      8.41
379 -3.40  0.60     11.56      0.36
380 -4.80 -2.60     23.04      6.76
381 -3.80  2.30     14.44      5.29
382 -1.60 -1.50      2.56      2.25
383 -4.30  1.00     18.49      1.00
384 -4.10  2.80     16.81      7.84
385 -1.60  0.20      2.56      0.04
386 -3.40 -0.80     11.56      0.64
387 -4.60  1.00     21.16      1.00
388 -4.70  0.70     22.09      0.49
389 -1.40 -0.10      1.96      0.01
390 -3.40 -0.70     11.56      0.49
391 -4.80 -0.10     23.04      0.01
392 -3.40  0.70     11.56      0.49
393 -1.50 -1.30      2.25      1.69
394 -4.70 -3.50     22.09     12.25
395 -2.60  0.40      6.76      0.16
396 -4.80  0.20     23.04      0.04
397 -3.30  1.70     10.89      2.89
398 -4.70 -0.00     22.09      0.00
399 -4.70  3.00     22.09      9.00

--------[ Data Set [500]---------------------------------------------


-----------------------------------------------------


+++++++++++++++++++++++++++SVM+++++++++++++++++++++++

C Parameter fine tuned vlaue for better accuracy  1
Weighted Vector :::: 0 500 {'Weight': array([[ 1.62, -1.66,  1.63, -1.62]]), 'intercept_': array([0.
04]), 'accuracy': 1.0}

-----------------------------------------------------
```

```
++++++++++++++++++++++++++++++++Percepron++++++++++++++++++

Weighted Vector :::: 500 {'Weight': array([[ 43.8 , -38.6 ,  45.56, -45.3 ]]), 'intercept_': array
([1.]), 'accuracy': 1.0}

-----------------------------------------------------


+++++++++++++++++++++ Kernel SVM  +++++++++++++++++++

========================================================accuracy 0.1 7 0.98
Weighted Vector :::: 500 {'Weight': 0, 'intercept_': array([0.05]), 'accuracy': 0.98}
```



Accuracy for SVM Kernel - size [500]

In [23]:
```
acc_list =[]
for i,w in enumerate(weight_SVM_Kernel_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for SVM Kernel ':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

Out[23]:

	Size of Data	Accuracy Calculation for SVM Kernel
0	500	0.98

Accuracy for SVM Linear - size [500]

In [24]:
```
acc_list =[]
for i,w in enumerate(weight_SVM_Linear_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for SVM Linear ':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

Out[24]:

	Size of Data	Accuracy Calculation for SVM Linear
0	500	1.00

Accuracy for Perceptron- size [500]

In [25]:
```
acc_list =[]
for i,w in enumerate(weight_perceptron_vector_list):
    acc_list.append(w['accuracy'])


d = { 'Size of Data ':size,'Accuracy Calculation for perceptron':acc_list}
df = pd.DataFrame(data=d)
df.head(len(size))
```

Out[25]:

	Size of Data	Accuracy Calculation for perceptron
0	500	1.00

In [ ]:

In [ ]:

In [ ]:

In [ ]: