```
1) Download the sample Python code that comes with the assigned textbook, Python Machine Learning.
```

# class Perceptron(object)

```
agitile class Perceptron(object)
import nummy as np
class Perceptron(object):

def __init__(self, eta=0.61, n_iter=50, random_state=1):
    self.cia = eta
    self.n_iter = n_iter
    self.n_ater = nadom_state
         def fit(self, X, y):
    rgen = np.random.RandomState(self.random_stateself.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
                   self.errors = []
for _in range(self.n[ter):
errors = 0
for xi, target in zip(X, y):
update = self.ets *(target - self.predict(xi))
self.w[1]:= update * xi
self.w[0] = update *
errors += int(update | x. 0)
self.errors _uppend(errors)
                   return self
         def net_input(self, X):
    """Calculate net input"""
    return np.dot(X, self.w_[i:]) + self.w_[0]
         def predict(self, X):
    return np.where(self.net_input(X) >= 0.0, 1, -1)
 class AdalineSGD(object)
 seif.random_state = random_state

def fit(self, X, y):
    self_initialize_weights(X.shape[1])
    self_initialize_weights(X.shape[1])
    for i in range(self.n.iter):
    if self.shuffle:
        X, y = self_shuffle(X, y)
    cost = []    self_initial(X, y):
        for al, tampet in siz(X, y):
        for al, tampet in siz(X, y):
        ang_cost = sum(cost) / len(y)
        self_cost_append(self_opter_weights(xi, tampet))
        return self
    return self
         def partial_fit(self, X, y):
    if not self.w.initialized:
        self._initialize_weights(X.shape[1])
    if y.ravel().shape[0] > 1:
        for x1, target in zip(X, y):
        self._update_weights(Xi, target)
                   self._update_weights(xi
else:
    self._update_weights(X, y)
return self
           def _shuffle(self, X, y):
    r = self.rgen.permutation(len(y))
    return X[r], y[r]
         - "INB

de judate wights(self, xi, target):

output = self.activation(self.net_input(xi))

error = (target - output)

self.w.[1:] = self.eta * xi.dot(error)

self.w.[0] + self.eta * error

cost = 0.5 * error*2

return cost
           \begin{array}{c} \text{def net\_input(self, X):} \\ \text{return np.dot(X, self.w}[1:]) + \text{self.w}[\theta] \end{array}
         def activation(self, X):
return X
         def predict(self, X):
    return np.where(self.activation(self.net_input(X))>= 0.0, 1, -1)
```

### plot\_decision\_regions() function

2) Use the Iris dataset that's referenced in the text

# Import Iris dataset

```
m@title Import Iris dataset
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df_iris = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
header=None)

        0
        1
        2
        3
        4

        145
        8.7
        3.0
        5.2
        2.3
        Iris-virginica

          146 6.3 2.5 5.0 1.9 Iris-virginica
         147 6.5 3.0 5.2 2.0 Iris-virginica
         148 6.2 3.4 5.4 2.3 Iris-virginica
         149 5.9 3.0 5.1 1.8 Iris-virginica
```

3) You can use the sample code from text as a starting point or you can write your own code from scratch.

4) Pick two classes of data (i.e., two species of Iris) and two features from the four in the dataset, so that the data for two species are lin separable using the features that you have chosen.

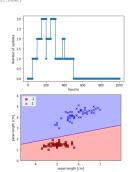
# TWO CLASS / TWO FEATURES - LINEARLY SEPARABLE

```
mptitle TWO CLASS / TWO FEATURES - LINEARLY SEPARABLE 
#Two species of iris: setota, versicolor 
#Two features from the data set: sepal length, petal length 
import pandas as pd 
import malpulib, pyplot as plt 
import numpy as op
df_iris.tail()
#pull two species of iris: setosa, versicolor y = df_iris.iloc[0:100, 4].values y = np.where(y == 'Iris-setosa', -1, 1)
```

```
2 1 45 50 55 60 sepal length [cm]
```

### TWO CLASS / TWO FEATURES - LINEARLY SEPARABLE: Perceptron

```
m@title TWO CLASS / TWO FEATURES - LINEARLY SEPARABLE: Perceptron import matplotlib.pyplot as plt import numpy as np from matplotlib.colors import ListedColormap
#applying Perceptron to two classes(setosa, versicolor), two features(sepal length, petal length)
pgn = Perceptron(eta=0.1, n_iter=10)
pgn = Ferceptron(eta=0.1, n_iter=10)
pgn.fit(X, y)
plt.plac(range(1, len(pgn.errors_) + 1), ppn.errors_, marker="o")
plt.vlabel('[optosis')
plt.vlabel('Subber of updates')
plt.vlabel('Subber of updates')
plt.show()
plt.savefig('lin_sep_perceptron_two_class_two_feat.png")
 #plot the decision regions:
plot_decision_regions(X, y, classifier=ppn)
plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc='upper left')
plt.show()
```

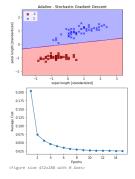


## TWO CLASS / TWO FEATURES - LINEARLY SEPARABLE: Adaline

```
#@title TWO CLASS / TWO FEATURES - LINEARLY SEPARABLE: Adaline import matplotlib.pyplot as plt
import map; as pit
import map; as pit
import map; as import instruction
import map; as import listedColormap
from matplotlib.colors import ListedColormap
sapplying Adalies to two classes(setosa, versicolor), two features(sepal length, petal length)
satundardize X (nage 43)
xxtd = por.cop(X)
X_std[:,0] = (X[:,0] - X[:,0].mean()) / X[:,0].std()
X_std[:,1] = (X[:,1] - X[:,1].mean()) / X[:,1].std()
ada = AdalineSGD(n_iter=15, eta=0.01, random_state=1) ada.fit(X_std, y)
```

plot\_decision\_regions(%\_std, y, classifier=ada) plt.title(\*Adaline - Stochastic Gradient Descent') plt.xlabel('speal length [standardized]') plt.ylabel('petal length [standardized]') plt.legend(loca'upper left') plt.show()

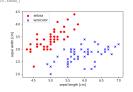
```
plt.plot(range(1, len(ada.cost_) + 1), ada.cost_, marker='o')
plt.xlabel('Epochs')
plt.ylabel('Average Cost')
plt.show()
plt.saver[g('lin_sep_adaline_two_class_two_feat.png')
```



5) Repeat Step 4) using three features at a time.

### TWO CLASS / THREE FEATURES - LINEARLY SEPARABLE

```
Spull three features from the data set: sepal length, sepal width, petal length x = df_1 ris_1 iloc(ge:100, (g.1,2]) values g splot data global spull scatter(<math>\chi(s;00,0), \chi(s;00,1), \chi(s;00,1),
```



```
Bapplying Perceptron to two classes(setona, versicolor), three features(sepal length, petal length, sepal width)
ppn = Perceptron(tei=0.1, n_iter=10)
ppn = Perceptron(tei=0.1, n_iter=10)
ppn = (Tit(X, y)
p
       TWO CLASS / THREE FEATURES - LINEARLY SEPARABLE: Adaline
   spiritle TWO CLASS / THREE FERTURES - LINEARLY SEPARABLE: Adaline import matplotlib.puplet as plt import mayes as no from antiportlib. Colors import ListedColormap from antiportlib. Colors import ListedColormap samplying Adaline to two classes(setosa, versicolor), three feature, x, \pm td: = n, copy(X). x, \pm td: = x, copy(X).
       ada = AdalineSGD(n_iter=15, eta=0.01, random_state=1) ada.fit(X_std, y)
       plt.plot(range(1, len(ada.cost_) + 1), ada.cost_, marker='o')
plt.ylabel('fepchs')
plt.ylabel('Average Cost')
plt.show()
plt.savefig("lin_sep_adaline_two_class_three_feat.png")
                                                                      0.175
                                                       0.100 -
                                            0.050 2 4 6 8 10 12 14

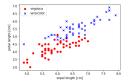
<Figure size 432x288 with 0 Axes>
       6) Repeat Step 4) using all four features at the same time.
           TWO CLASS / FOUR FEATURES - LINEARLY SEPARABLE
       mgtitle TWO CLASS / FOUR FEATURES - LINEARLY SEPARABLE mpull two species of iris: setosa, versicolor y = df_iris.iloc[0:100, 4].values y = np.where(y == 'Iris-setosa', -1,1)
       spull four features from the data set: sepal length, sepal width, petal length, petal width x = df_{\pm}^{*}i\pi_{\pm}i.inc(g:100, [0,1,2,3]),values a plot data pli.scatter(X[:50, 0], X[:50, 1], color='red', marker-to', label='setosa')
   plt.scatter(X[50:180, 0], X[50:180, 1], color-'blue', narker-'a', label-'versicolor') plt.ylabel('spal length (cal') plt.ylabel('spal width (cal') splt.label('spal width (cal') plt.label('spal length (cal') plt.label('portlength (cal') plt.label('p
                                                           To set of the set of t
       TWO CLASS / FOUR FEATURES - LINEARLY SEPARABLE: Perceptron
       #applying Perceptron to two classes(setosa, ve
                                                                                                                                                                                                                                                                                                                                                                                                                            rsicolor), four features(sepal length, petal length, sepal width, petal width)
   ppn = Perceptron(eta=0.1, n_iter=18)
ppn.fit(X, y)
ppn.fit(X, y)
plt.jolf(ramge(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.jolae(| ramge(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
ppn.errors_, marker=
       TWO CLASS / FOUR FEATURES - LINEARLY SEPARABLE: Adaline
   mgittle TwO CLASS / FOUR FEATURES - LINEARLY SEPARABLE: Adaline
import satplatilb.pyplot as plt
import manys as no
from matplatilb.colors import ListedColormap
mapplying Adaline to bus classes(setosa, versicolor), four features(sepal length, petal length, sepal width, petal width)
%_std = np.copy(X)
%_std[:,1] = (X[:,0] = X[:,0] = men()) / X[:,0]:,std()
%_std[:,1] = (X[:,1] = X[:,1].mean()) / X[:,1].std()
       \label{eq:ada} \begin{array}{lll} ada = AdalineSGD(n\_iter=15,\ eta=0.01,\ random\_state=1) \\ ada.fit(X\_std,\ y) \end{array}
       plt.plot(range(1, len(ada.cost_) + 1), ada.cost_, marker='o')
plt.vlabel('Epochs')
plt.vlabel('Average Cost')
plt.show()
plt.savefig('lin_sep_adaline_two_class_four_feat.png')
                                                                      0.175
                                                       0.150 of 0.125 of 0.1
```

2 4 6 8 10 12 14 Epochs <Figure size 432x288 with 0 Axes>

7) Pick two classes of data (i.e., two species of fris) and two features from the four in the dataset, so that the data for two species are NOT linearly separable using the features that you have chosen.

### TWO CLASS / TWO FEATURES - NOT LINEARLY SEPARABLE

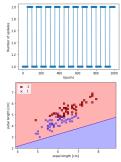
```
Hightile TWO CLASS / TWO FEATURES - NOT LIMEARLY SEPARABLE Relaxes: Virginics and verticolor, features: sepal length, petal length import pands as import pands as paid import pands as paid import astallation of the properties of
   df_iris * pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', header=None) df_iris.tail()
      #pull two species of iris: virginica, versicolor
y = df_iris.iloc[50:150, 4].values
y = np.where(y == 'Iris-virginica', -1, 1)
   pull two features from the data set: sepal length, petal length X = df_iris.ilo[[9:150, [0,2]].values # plot data plt.scatter([:50, 0], X[:50, 1], color='red', marker='o', label='virginica')
namewo', labela'vinginica')
pit.scatter(X[58:180, 0], X[50:180, 1],
color-'blue', namewo', labela'versicolor')
pit.xlabel('sepal length [cm]')
pit.ylabel('setal length [cm]')
pit.ylabel('sepal length [cm]')
pit.sbee()
```



Then Apply the Perceptron and Adaline models to the classes/features that you have chosen and report your results.

### TWO CLASS / TWO FEATURES - NOT LINEARLY SEPARABLE: Perceptron

```
m@title TWO CLASS / TWO FEATURES - NOT LINEARLY SEPARABLE: Perceptron import matplotlib.pyplot as plt import numpy as np from matplotlib.colors import ListedColormap
#applying Perceptron to two classes(virginica, versicolor), two features(sepal length, petal length)
pm = Perceptron(eta=0.1, n_ite=10)
pm = Perceptron(eta=0.1, n_ite=10)
pm = Perceptron(eta=0.1, n_ite=10)
pm = Perceptron(eta=0.1, n_ite=10)
pht.iolet("Epochs")
```

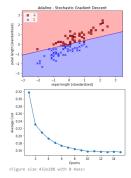


# TWO CLASS / TWO FEATURES - NOT LINEARLY SEPARABLE: Adaline

```
#@title TWO CLASS / TWO FEATURES - NOT LINEARLY SEPARABLE: Adaline import matplotlib.pyplot as plt
isport satploils.psplot as plt isport satploils.psplot as plt isport samples for size for size of the format plane in the format plane is format plane in the format plane is possible for size for size
    \label{eq:ada} \begin{array}{lll} \text{ada} = & \text{AdalineSGD}(\text{n\_iter=15, eta=0.01, random\_state=1}) \\ \text{ada.fit}(\text{X\_std, y}) \end{array}
```

plot\_decision\_regions(X\_std, y, classifier=ada)
plt.title('Adaline - Stochastic Gradient Descent')
plt.xlabel('speal lengt [standardized]')
plt.ylabel('petal length [standardized]')
plt.legend(loc='upper left')
plt.show()

plt.plot(range(1, len(ada.cost\_) + 1), ada.cost\_, marker='o')
plt.xlabel('kpcnb')
plt.ylabel('Average Cost')
plt.show()
plt.savefig('non\_lin\_sep\_adaline\_two\_class\_two\_feat.png")



8) Repeat Step 7) using three features at a time.

### TWO CLASS / THREE FEATURES - NOT LINEARLY SEPARABLE

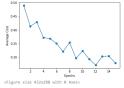
```
sgtitle TWO CLASS / THREE FEATURES - NOT LINEARLY SEPARABLE
spull two species of iris: virginica, versicolor
y = of_iris.iloo[Se:150, 4].values
y = np.where(y == 'Iris-virginica', -1, 1)
Spull three features from the data set: sepal length, sepal width, petal length X = df_1 fris.iloc(Soi:S0, [0,1,2]).values a plot data a plot data plt.scatter(<math>X[:S0, 0], X[:S0, 1], color="red", marker-to", label='virginica')
```

```
6.0 6.5 7.0
sepal length [cm]
  TWO CLASS / THREE FEATURES - NOT LINEARLY SEPARABLE: Perceptron
  mBtitle TWO CLASS / THREE FEATURES - MOT LINEARLY SEPARABLE: Perce import matplotlib.pyplot as plt import numpy as np from matplotlib.colors import ListedColormap
 #applying Perceptron to two classes(virginica, versicolor), three features(sepal length, petal length, sepal width)
ppn = Perceptron(eta=0.1, n_iter=10)
ppn :#IXI, yp
plt.plef(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.ylabel('subber o' updates')
plt.ylabel('subber of updates')
plt.label('subber of updates')
plt.label('plt.savefig('non_lin_sep_perceptron_two_class_three_feat.png')
                  sate 16
  TWO CLASS / THREE FEATURES - NOT LINEARLY SEPARABLE: Adaline
   #@title TWO CLASS / THREE FEATURES - NOT LINEARLY SEPARABLE: Adaline import matplotlib.pyplot as plt
 isport amplied as pit isport maps as more import insight as for size in the factor maps as for import maps as form is given this decidence in the factor of the factor of
   \label{eq:ada} \begin{array}{lll} \text{ada} = & \text{AdalineSGD}(\text{n\_iter=15, eta=0.01, random\_state=1}) \\ \text{ada.fit}(\text{X\_std, y}) \end{array}
 plt.plot(range(1, len(ada.cost_) + 1), ada.cost_, marker='o')
plt.vlabel('Aperage Cost')
plt.ylabel('Average Cost')
plt.show()
plt.savefig('non_lin_sep_adaline_two_class_three_feat.png')
  9) Repeat Step 7) using all four features at the same time.
  TWO CLASS / FOUR FEATURES - NOT LINEARLY SEPARABLE
  m@title TWO CLASS / FOUR FEATURES - NOT LINEARLY SEPARABLE
mpull two species of iris: virginica, versicolor
y = off-iris.loc[$0:150, 4].values
y = np.where(y == 'Iris-virginica', -1,1)
 TWO CLASS / FOUR FEATURES - NOT LINEARLY SEPARABLE: Perceptron
  m@title TMO CLASS / FOUR FEATURES - NOT LINEARLY SEPARABLE: Perceptron import matplotlib.pyplot as plt import numpy as np from matplotlib.colors import ListedColormap
 #applying Perceptron to two classes(virginica, versicolor), four features(sepal length, sepal width, petal length, petal width)
pm = Perceptron(eta=0.1, n_iter=18)
pm.fif(x);
plt.plack(reage(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.plack(reage(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.plack('Number of updates')
plt.plack('Number of updates')
plt.lawefig('non_lin_sep_perceptron_two_class_four_feat.png')
                   16 -
                0 200 400 600
Epochs
<Figure size 432x288 with 0 Axes>
  TWO CLASS / FOUR FEATURES - NOT LINEARLY SEPARABLE: Adaline
mgittle TwO CLASS / FOUR FERINRES - NOT INMERRY SEPARABLE: Adaline
import matple as plt
import many as no
from matpletlib.colors import ListedColormap
mapplying Adaline to bu classes(virginica, versicolor), four features(sepal length, sepal width, petal length, petal width)
%_xtd = no.copy(3)
%_xtd[:,a] = (M[:,a] - M[:,a], seen()) / M[:,a];std()
%_xtd[:,a] = (M[:,a] - M[:,a], seen()) / M[:,a];std()
```

i00, 1], r='x', label='versicolor')

ada = AdalineSGD(n\_iter=15, eta=0.01, random\_state=1)
ada.fit(X\_std, y)

plt.plot(range(1, len(ada.cost\_) + 1), ada.cost\_, marker=',



10) Compare your results for the Perceptron and Adaline models.

All applications of the Adaline models on the non linearly separable data ran with fewer epochs. The perceptron models ran with a high number of epochs on the non linearly separable data.

11) Submit your results as an ipython notebook (i.e. in ipynb format).

✓ 0s completed at 7:51 PM