

Name: Venkata Krishnarjun Vuppala

SRN: PES2UG19CS451

Semester:6

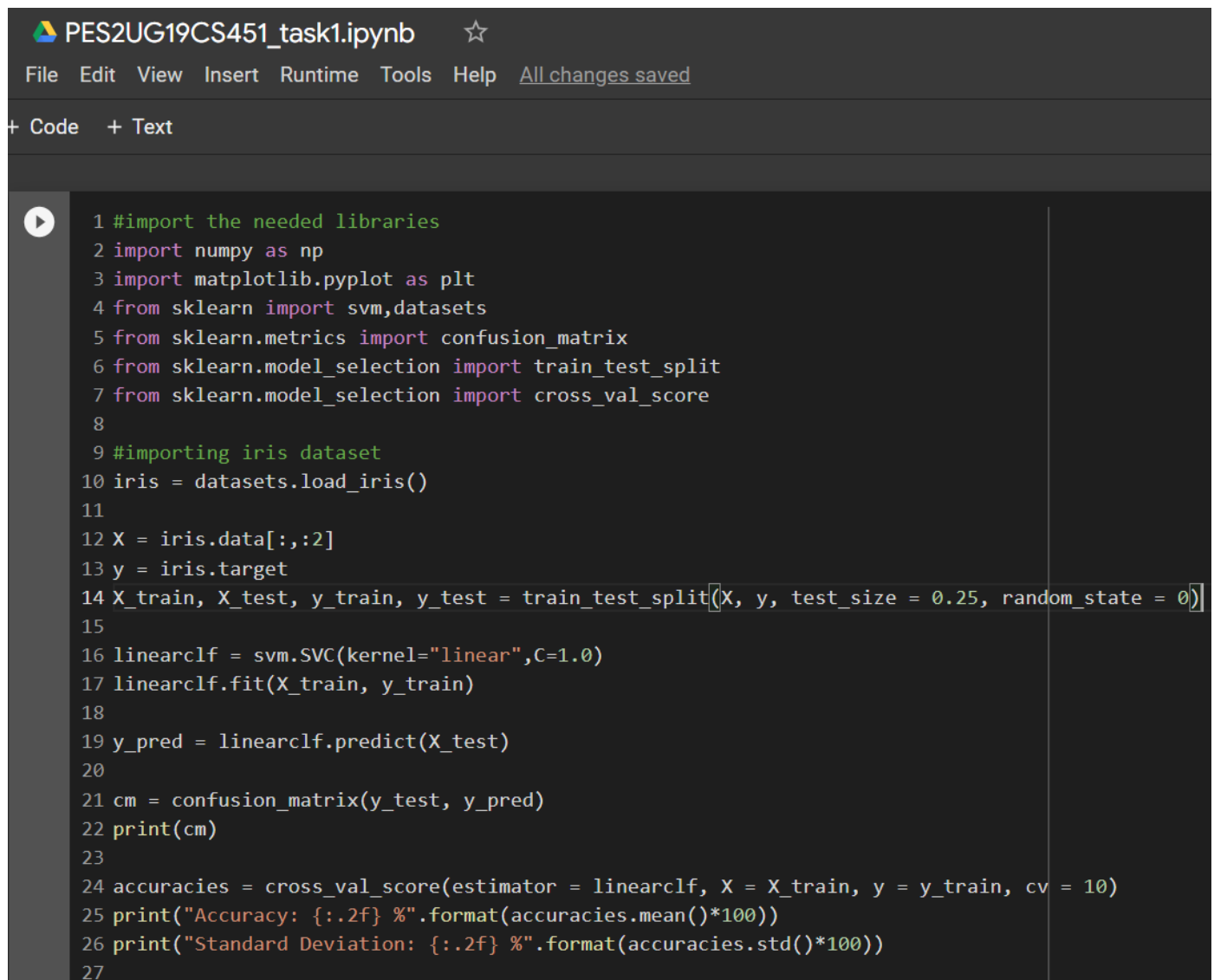
Section: G

Subject: Topics in Deep Learning

Assignment 2

Output Screenshots:

1. SVM - Linear classification



```
1 #import the needed libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn import svm, datasets
5 from sklearn.metrics import confusion_matrix
6 from sklearn.model_selection import train_test_split
7 from sklearn.model_selection import cross_val_score
8
9 #importing iris dataset
10 iris = datasets.load_iris()
11
12 X = iris.data[:, :2]
13 y = iris.target
14 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
15
16 linearclf = svm.SVC(kernel="linear", C=1.0)
17 linearclf.fit(X_train, y_train)
18
19 y_pred = linearclf.predict(X_test)
20
21 cm = confusion_matrix(y_test, y_pred)
22 print(cm)
23
24 accuracies = cross_val_score(estimator = linearclf, X = X_train, y = y_train, cv = 10)
25 print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
26 print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
27
```



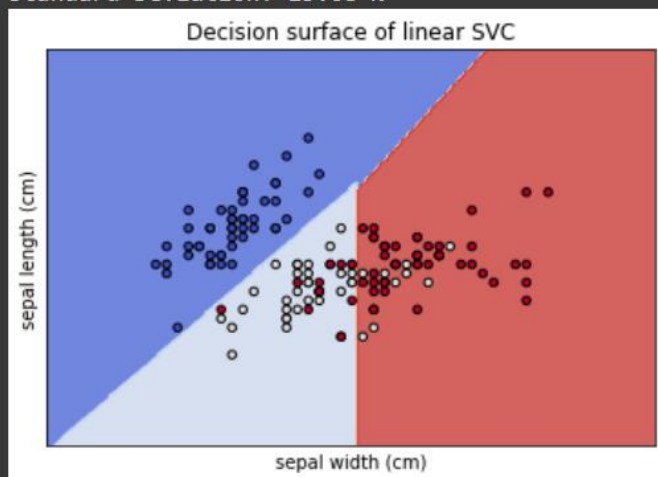
```
28 def make_meshgrid(x, y, h=.02):
29     x_min, x_max = x.min()-1, x.max() + 1
30     y_min, y_max = y.min()-1, y.max() + 1
31     xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
32     return xx, yy
33
34 def plot_contours(ax, clf, xx, yy, **params):
35     Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
36     Z = Z.reshape(xx.shape)
37     out = ax.contourf(xx, yy, Z, **params)
38     return out
39
40 feature_names = iris.feature_names[:2]
41 classes = iris.target_names
42 fig, ax = plt.subplots()
43 title = ('Decision surface of linear SVC')
44 X0, X1 = X[:, 0], X[:, 1]
45 xx, yy = make_meshgrid(X0, X1)
46 plot_contours(ax, linearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
47 ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
48 ax.set_ylabel("{}".format(feature_names[0]))
49 ax.set_xlabel("{}".format(feature_names[1]))
50 ax.set_xticks(())
51 ax.set_yticks(())
52 ax.set_title(title)
53 plt.show()
```



```
[[13  0  0]
 [ 0 11  5]
 [ 0  4  5]]
```

Accuracy: 81.29 %

Standard Deviation: 13.68 %



2. SVM - Non-Linear classification

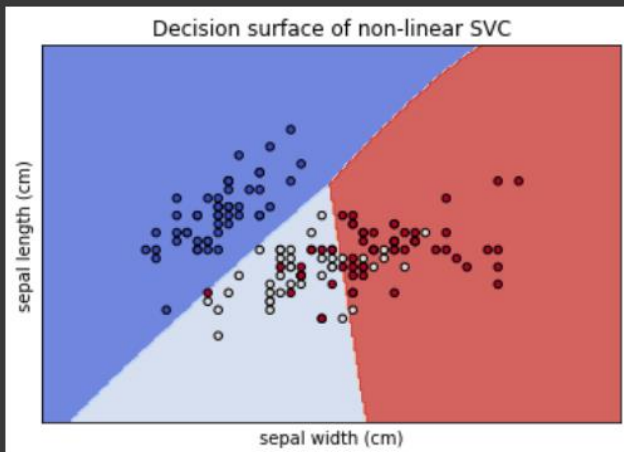
PES2UG19CS451_task2.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Code + Text

```
29 nonLinearclf.fit(X_train, y_train)
30
31 y_pred = nonLinearclf.predict(X_test)
32
33 cm = confusion_matrix(y_test, y_pred)
34 print(cm)
35
36 accuracies = cross_val_score(estimator = nonLinearclf, X = X_train, y = y_train, cv = 10)
37 print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
38 print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
39
40 feature_names = iris.feature_names[:2]
41 classes = iris.target_names
42 fig, ax = plt.subplots()
43 title = ('Decision surface of non-linear SVC')
44 X0, X1 = X[:, 0], X[:, 1]
45 xx, yy = make_meshgrid(X0, X1)
46 plot_contours(ax, nonLinearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
47 ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
48 ax.set_ylabel("{} {}".format(feature_names[0]))
49 ax.set_xlabel("{} {}".format(feature_names[1]))
50 ax.set_xticks(())
51 ax.set_yticks(())
52 ax.set_title(title)
53 plt.show()
```

```
[[13  0  0]
 [ 0 11  5]
 [ 0  4  5]]
Accuracy: 82.20 %
Standard Deviation: 12.75 %
```



3. To implement SVM for IRIS dataset using linear kernel/ RBF kernel.

```
PES2UG19CS451_task3.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 4:38 PM
+ Code + Text

[2] 1 #import the needed libraries
    2 import numpy as np
    3 import matplotlib.pyplot as plt
    4 from sklearn.metrics import mean_squared_error, confusion_matrix, precision_score, recall_score, auc, roc_curve
    5 import pandas as pd
    6 from sklearn import svm, datasets
    7 from sklearn.metrics import confusion_matrix
    8 from sklearn import model_selection
    9 from sklearn.model_selection import train_test_split
   10 from sklearn.model_selection import cross_val_score
   11
   12 #importing iris dataset
   13 iris = datasets.load_iris()
   14
   15 X = iris.data[:, :2]
   16 y = iris.target
   17 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
   18
   19 linearclf = svm.SVC(kernel="linear", C=1.0)
   20 linearclf.fit(X_train, y_train)
   21
   22 y_pred = linearclf.predict(X_test)
   23
   24 cm = confusion_matrix(y_test, y_pred)
   25 print(cm)
   26
   27 accuracies = cross_val_score(estimator = linearclf, X = X_train, y = y_train, cv = 10)
   28 print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
```

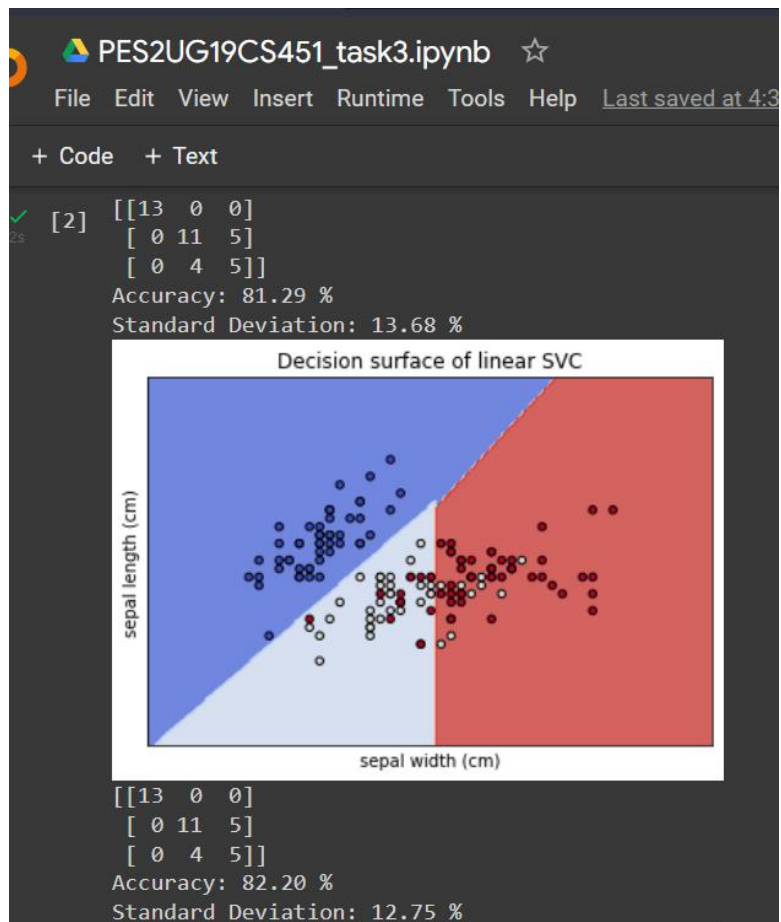
+ Code + Text

```
[2] 29 print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
    30
    31 def make_meshgrid(x, y, h=.02):
    32     x_min, x_max = x.min()-1, x.max() + 1
    33     y_min, y_max = y.min()-1, y.max() + 1
    34     xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    35     return xx, yy
    36
    37 def plot_contours(ax, clf, xx, yy, **params):
    38     Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    39     Z = Z.reshape(xx.shape)
    40     out = ax.contourf(xx, yy, Z, **params)
    41     return out
    42
    43 feature_names = iris.feature_names[:2]
    44 classes = iris.target_names
    45 fig, ax = plt.subplots()
    46 title = ('Decision surface of linear SVC')
    47 X0, X1 = X[:, 0], X[:, 1]
    48 xx, yy = make_meshgrid(X0, X1)
    49 plot_contours(ax, linearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
    50 ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
    51 ax.set_ylabel("{} ".format(feature_names[0]))
    52 ax.set_xlabel("{} ".format(feature_names[1]))
    53 ax.set_xticks(())
    54 ax.set_yticks(())
    55 ax.set_title(title)
    56 plt.show()
```

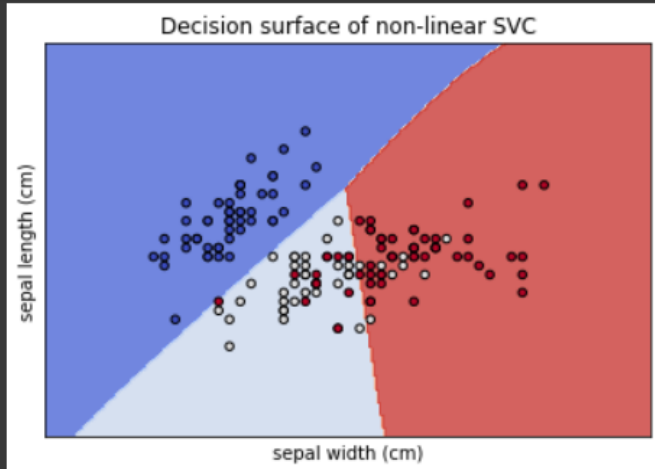
+ Code + Text

```
[2] 58 nonLinearclf = svm.SVC(kernel="rbf",C=1.0)
59 nonLinearclf.fit(X_train, y_train)
60
61 y_pred = nonLinearclf.predict(X_test)
62
63 cm = confusion_matrix(y_test, y_pred)
64 print(cm)
65
66 accuracies = cross_val_score(estimator = nonLinearclf, X = X_train, y = y_train, cv = 10)
67 print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
68 print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
69
70 fig, ax = plt.subplots()
71 title = ('Decision surface of non-linear SVC')
72 X0, X1 = X[:, 0], X[:, 1]
73 xx, yy = make_meshgrid(X0, X1)
74 plot_contours(ax, nonLinearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
75 ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
76 ax.set_ylabel("{} ".format(feature_names[0]))
77 ax.set_xlabel("{} ".format(feature_names[1]))
78 ax.set_xticks(())
79 ax.set_yticks(())
80 ax.set_title(title)
81 plt.show()
82
83 models = []
84 results = []
85 names = []
86 seed = 7
87 models.append(('SVM with Linear Kernel', svm.SVC(kernel="linear",C=1.0)))
88 models.append(('SVM with non-Linear Kernel', svm.SVC(kernel="rbf",C=1.0)))
89 scoring = 'accuracy'
```

```
88 models.append(('SVM with non-Linear Kernel', svm.SVC(kernel="rbf",C=1.0)))
89 scoring = 'accuracy'
90 for name, model in models:
91     kfold = model_selection.KFold(n_splits=10,shuffle=True, random_state=seed)
92     cv_results = model_selection.cross_val_score(model, X, y, cv=kfold, scoring=scoring)
93     results.append(cv_results)
94     names.append(name)
95     msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
96     print(msg)
97
98 fig = plt.figure()
99 fig.suptitle('Algorithm Comparison')
100 ax = fig.add_subplot(111)
101 plt.boxplot(results)
102 ax.set_xticklabels(names)
103 plt.show()
```



[2]



SVM with Linear Kernel: 0.780000 (0.111754)

SVM with non-Linear Kernel: 0.813333 (0.083267)

