Name: Venkata Krishnarjun Vuppala **Semester**:6
SRN: PES2UG19CS451 **Section**: G

Subject: Topics in Deep Learning

Assignment 2

Output Screenshots:

1. SVM - Linear classification

PES2UG19CS451_task1.ipynb

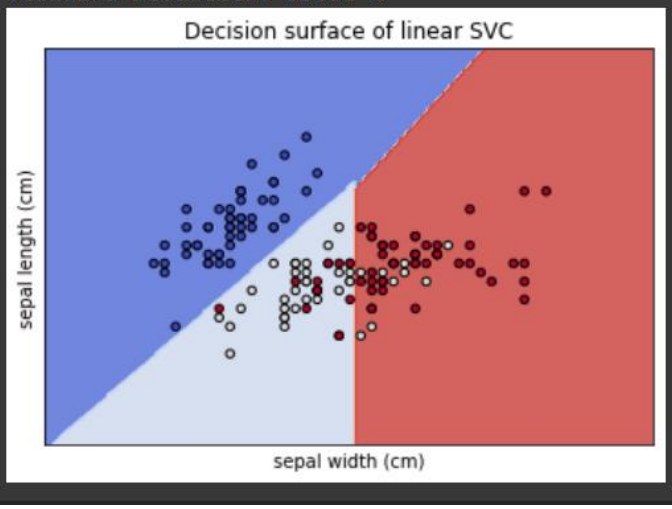
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
#import the needed libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm,datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

#importing iris dataset
iris = datasets.load_iris()

X = iris.data[:,:2]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

linearclf = svm.SVC(kernel="linear",C=1.0)
linearclf.fit(X_train, y_train)

y_pred = linearclf.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracies = cross_val_score(estimator = linearclf, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

```

```python
def make_meshgrid(x, y, h=.02):
    x_min, x_max = x.min()-1, x.max() + 1
    y_min, y_max = y.min()-1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out

feature_names = iris.feature_names[:2]
classes = iris.target_names
fig, ax = plt.subplots()
title = ('Decision surface of linear SVC')
X0, X1 = X[:, 0], X[:, 1]
xx, yy = make_meshgrid(X0, X1)
plot_contours(ax, linearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
ax.set_ylabel("{}".format(feature_names[0]))
ax.set_xlabel("{}".format(feature_names[1]))
ax.set_xticks(())
ax.set_yticks(())
ax.set_title(title)
plt.show()
```

```
[[13  0  0]
 [ 0 11  5]
 [ 0  4  5]]
Accuracy: 81.29 %
Standard Deviation: 13.68 %
```



2. SVM - Non-Linear classification

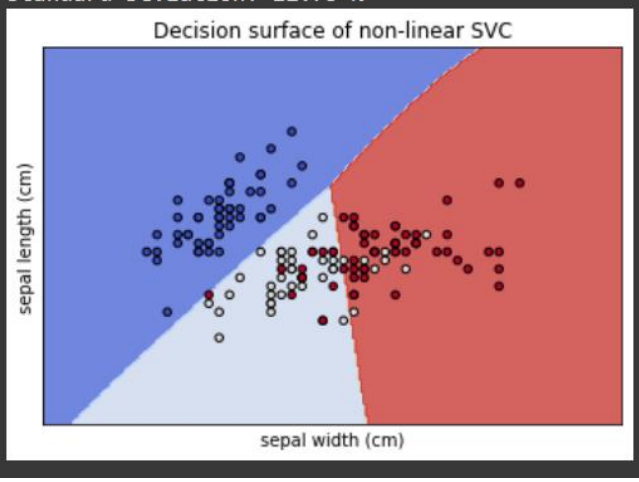
PES2UG19CS451_task2.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
nonLinearclf.fit(X_train, y_train)

y_pred = nonLinearclf.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracies = cross_val_score(estimator = nonLinearclf, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

feature_names = iris.feature_names[:2]
classes = iris.target_names
fig, ax = plt.subplots()
title = ('Decision surface of non-linear SVC')
X0, X1 = X[:, 0], X[:, 1]
xx, yy = make_meshgrid(X0, X1)
plot_contours(ax, nonLinearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
ax.set_ylabel("{}".format(feature_names[0]))
ax.set_xlabel("{}".format(feature_names[1]))
ax.set_xticks(())
ax.set_yticks(())
ax.set_title(title)
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]

```
#import the needed libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error,confusion_matrix, precision_score, recall_score, auc,roc_curve
import pandas as pd
from sklearn import svm,datasets
from sklearn.metrics import confusion_matrix
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

#importing iris dataset
iris = datasets.load_iris()

X = iris.data[:,:2]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

linearclf = svm.SVC(kernel="linear",C=1.0)
linearclf.fit(X_train, y_train)

y_pred = linearclf.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracies = cross_val_score(estimator = linearclf, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
```

PES2UG19CS451_task3.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 4:38 PM

+ Code + Text

[2]

```
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

def make_meshgrid(x, y, h=.02):
    x_min, x_max = x.min()-1, x.max() + 1
    y_min, y_max = y.min()-1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out

feature_names = iris.feature_names[:2]
classes = iris.target_names
fig, ax = plt.subplots()
title = ('Decision surface of linear SVC')
X0, X1 = X[:, 0], X[:, 1]
xx, yy = make_meshgrid(X0, X1)
plot_contours(ax, linearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
ax.set_ylabel("{}".format(feature_names[0]))
ax.set_xlabel("{}".format(feature_names[1]))
ax.set_xticks(())
ax.set_yticks(())
ax.set_title(title)
plt.show()
```

PES2UG19CS451_task3.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 4:38 PM

+ Code + Text

[2]

```
nonLinearclf = svm.SVC(kernel="rbf",C=1.0)
nonLinearclf.fit(X_train, y_train)

y_pred = nonLinearclf.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracies = cross_val_score(estimator = nonLinearclf, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

fig, ax = plt.subplots()
title = ('Decision surface of non-linear SVC')
X0, X1 = X[:, 0], X[:, 1]
xx, yy = make_meshgrid(X0, X1)
plot_contours(ax, nonLinearclf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
ax.set_ylabel("{}".format(feature_names[0]))
ax.set_xlabel("{}".format(feature_names[1]))
ax.set_xticks(())
ax.set_yticks(())
ax.set_title(title)
plt.show()

models = []
results = []
names = []
seed = 7
models.append(('SVM with Linear Kernel', svm.SVC(kernel="linear",C=1.0)))
models.append(('SVM with non-Linear Kernel', svm.SVC(kernel="rbf",C=1.0)))
scoring = 'accuracy'
```

```
models.append(('SVM with non-Linear Kernel', svm.SVC(kernel="rbf",C=1.0)))
scoring = 'accuracy'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10,shuffle=True, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X, y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

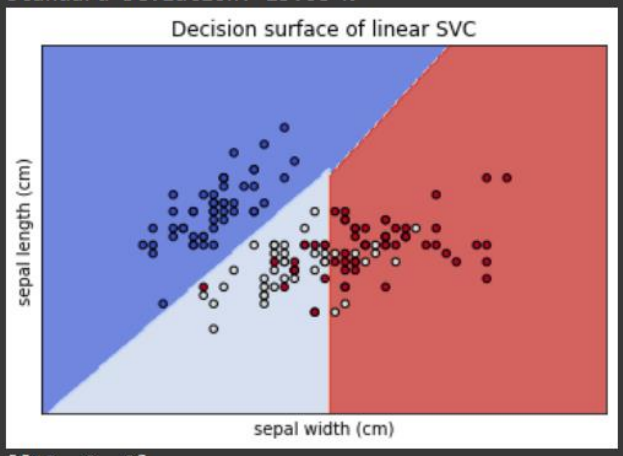
PES2UG19CS451_task3.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 4:3

+ Code + Text

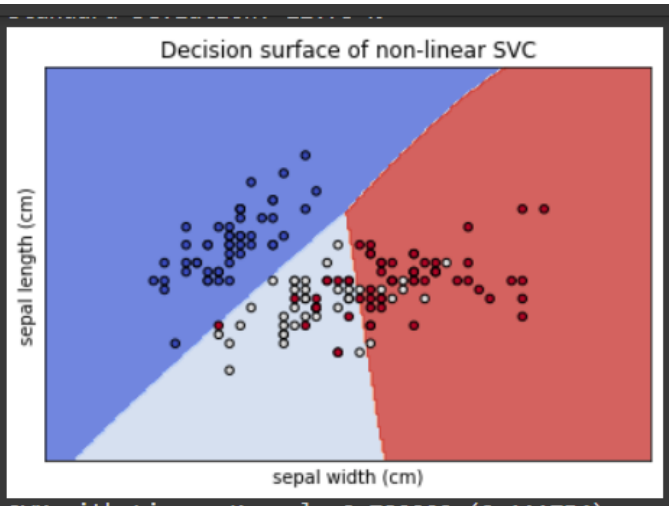
[2]

```
[[13  0  0]
 [ 0 11  5]
 [ 0  4  5]]
Accuracy: 81.29 %
Standard Deviation: 13.68 %
```




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

[2]




```
SVM with Linear Kernel: 0.780000 (0.111754)
SVM with non-Linear Kernel: 0.813333 (0.083267)
```