SVM

# Coding Assignment 6[¶](#Coding-Assignment-6)

SVM  
Dataset: Pima Indians Diabetes Database  
Author: Sreejith S

Dataset: <https://www.kaggle.com/uciml/pima-indians-diabetes-database>

In [21]:

import numpy as np  
import matplotlib.pyplot as plt   
import pandas as pd  
  
%reload\_ext autoreload  
%matplotlib inline  
%autoreload 2  
%config InlineBackend.figure\_format = 'retina'  
  
  
#classifiers  
from sklearn.dummy import DummyClassifier  
from sklearn import svm  
  
from sklearn.model\_selection import train\_test\_split  
  
#evaluation metrics  
from sklearn.metrics import plot\_confusion\_matrix, confusion\_matrix, ConfusionMatrixDisplay  
from sklearn.metrics import plot\_roc\_curve, roc\_curve, roc\_auc\_score   
from sklearn.metrics import f1\_score  
from sklearn.metrics import plot\_precision\_recall\_curve, precision\_recall\_curve, precision\_score, recall\_score  
from sklearn.metrics import matthews\_corrcoef, average\_precision\_score  
  
  
#plotting utils  
from utils import plot\_table, plot\_confusion

In [40]:

#set pd display options  
pd.set\_option('display.max\_columns', 10)  
pd.set\_option('display.width', 80)  
  
# Importing the dataset  
dataset = pd.read\_csv('../datasets/pima-indians-diabetes.csv')   
  
#selecting features  
f\_best = ['plasma\_glucose', 'serum\_insulin', 'bmi', 'diab\_pedigree']  
f\_a = ['age', 'pregnant', 'bmi']  
f\_b = ['skin\_thickness', 'diab\_pedigree', 'dia\_BP']  
f\_all = ['pregnant', 'plasma\_glucose', 'dia\_BP', 'skin\_thickness', 'serum\_insulin', 'bmi', 'diab\_pedigree', 'age']  
  
def test\_model(classifiers, feature\_sets, short\_titles, titles):  
 """  
 Arguments:  
 classifiers - list of classifiers  
 feature\_sets - list of featuresets to be used  
 """  
  
 y = dataset[['Diab']].values.ravel()  
 confusion\_matrices = []  
 eval\_metrics\_list = []  
  
  
 fig = plt.figure(figsize=(10, 20))  
 plt.style.use("seaborn-whitegrid")  
 ax\_roc = fig.add\_subplot(3, 1, 1)  
 ax\_roc.set\_title("Receiver Operating Characteristic Curve")  
 ax\_pre = fig.add\_subplot(3, 1, 2)  
 ax\_pre.set\_title("Precision-Recall Curve")  
  
 ax\_table = fig.add\_subplot(3, 1, 3)  
 ax\_table.set\_title("Evaluation Metrices")  
  
  
 for i, (clf, feature) in enumerate(zip(classifiers, feature\_sets)):  
 X = dataset[feature]  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,   
 random\_state=0)  
 clf.fit(X\_train, y\_train)  
 y\_preds = clf.predict(X\_test)  
 y\_proba = clf.predict\_proba(X\_test)[:, 1]  
  
 plot\_roc\_curve(clf, X\_test, y\_test, ax=ax\_roc, name=titles[i])   
 plot\_precision\_recall\_curve(clf, X\_test, y\_test, ax=ax\_pre, name=titles[i])  
  
 confusion\_matrices.append(confusion\_matrix(y\_test, y\_preds))  
 eval\_metrics = [clf.score(X\_test, y\_test),  
 precision\_score(y\_test, y\_preds, zero\_division=0),  
 recall\_score(y\_test, y\_preds),  
 f1\_score(y\_test, y\_preds),  
 matthews\_corrcoef(y\_test, y\_preds),  
 roc\_auc\_score(y\_test, y\_proba),  
 average\_precision\_score(y\_test, y\_proba)]  
 eval\_metrics = [f"{i:.2f}" for i in eval\_metrics]  
 eval\_metrics\_list.append(eval\_metrics)  
  
 ax\_pre.legend(loc='upper right')  
 plot\_table(eval\_metrics\_list, ax\_table, short\_titles)  
 plot\_confusion(confusion\_matrices, short\_titles)  
 plt.show()

### 6.1. [1 marks] Create a dummy classifier (see model evaluation video from AnacondaCon), fit it using the three featuresets (a) all features, (b) Only age, pregnant, bmi and (c) only skin\_thickness, diab\_pedigree, and dia\_BP as the input features.[¶](#Xd81972ff3043c514d3654293a281a994c0b7d53)

Evaluate each of these above models using the metrics of precision, recall, F1-score, and AUROC (area under receiver operating characteristics).

In [41]:

short\_titles = ["DC-4", "DC-a", "DC-b", "DC-all"]  
titles = ["DummyClf - plasma\_glucose, serum\_insulin, bmi, pedigree",   
 "DummyClf - age, #preg, bmi",   
 "DummyClf - skin\_th, pedigree, B.P",   
 "DummyClf - All features"]  
feature\_sets = [f\_best, f\_a, f\_b, f\_all]  
  
dummy = DummyClassifier(strategy='most\_frequent', random\_state=0)  
classifiers = [dummy, dummy, dummy, dummy]  
test\_model(classifiers, feature\_sets, short\_titles, titles)

/home/sj/.local/lib/python3.8/site-packages/sklearn/metrics/\_classification.py:846: RuntimeWarning: invalid value encountered in double\_scalars  
 mcc = cov\_ytyp / np.sqrt(cov\_ytyt \* cov\_ypyp)  
/home/sj/.local/lib/python3.8/site-packages/sklearn/metrics/\_classification.py:846: RuntimeWarning: invalid value encountered in double\_scalars  
 mcc = cov\_ytyp / np.sqrt(cov\_ytyt \* cov\_ypyp)  
/home/sj/.local/lib/python3.8/site-packages/sklearn/metrics/\_classification.py:846: RuntimeWarning: invalid value encountered in double\_scalars  
 mcc = cov\_ytyp / np.sqrt(cov\_ytyt \* cov\_ypyp)  
/home/sj/.local/lib/python3.8/site-packages/sklearn/metrics/\_classification.py:846: RuntimeWarning: invalid value encountered in double\_scalars  
 mcc = cov\_ytyp / np.sqrt(cov\_ytyt \* cov\_ypyp)

![](data:image/png;base64;base64,)

![](data:image/png;base64;base64,)

### 6.2. [3 marks] Write the code to build an SVM classification model with linear kernel. Use training data to fit and test data to evaluate. For this question, build the model using all features.[¶](#X1f2c79529b12899a3f918d32f785f3e9f76a320)

Evaluate the SVM model that you have created using the metrics of precision, recall, F1-score, and AUROC (area under receiver operating characteristics), and tabulate the results below.

In [32]:

short\_titles = ["SVM\_Lin-all", "SVM\_Lin-best"]  
titles = ["SVM Linear - All features", "SVM Linear - Best 4 features"]  
feature\_sets = [f\_all, f\_best]  
  
svm\_clf = svm.SVC(kernel='linear', probability=True) #by default probability is disabled  
classifiers = [svm\_clf, svm\_clf]  
test\_model(classifiers, feature\_sets, short\_titles, titles)

f: ['pregnant', 'plasma\_glucose', 'dia\_BP', 'skin\_thickness', 'serum\_insulin', 'bmi', 'diab\_pedigree', 'age']  
f: ['plasma\_glucose', 'serum\_insulin', 'bmi', 'diab\_pedigree']

![](data:image/png;base64;base64,)

![](data:image/png;base64;base64,)

### 6.3. [2 marks] Write code to build SVM classification models with ‘rbf’ kernel and ‘poly’ kernel. Use training data to fit and test data to evaluate. For this question, build the model using all features. Evaluate the SVM model that you have created using the metrics of precision, recall, F1-score, and AUROC (area under receiver operating characteristics), and tabulate the results below.[¶](#Xe70ff310702b7f205d9f9e23a9e1c128d7394a3)

In [26]:

short\_titles = ["SVM\_rbf-all", "SVM\_poly-all"]  
titles = ["SVM rbf - All features", "SVM poly - All features"]  
feature\_sets = [f\_all, f\_all]  
  
svm\_clf\_rbf = svm.SVC(kernel='rbf', probability=True)  
svm\_clf\_poly = svm.SVC(kernel='poly', probability=True)  
  
classifiers = [svm\_clf\_rbf, svm\_clf\_poly]  
test\_model(classifiers, feature\_sets, short\_titles, titles)

clf: 140203354633360  
clf: 140203295462640

![](data:image/png;base64;base64,)

![](data:image/png;base64;base64,)

### 6.4. [3 marks] Write code to build SVM classification models with ‘linear’, ‘rbf’ and ‘poly’ kernels using input featuresets as: (a) Only age, pregnant, bmi and (b) only skin\_thickness, diab\_pedigree, and dia\_BP. Use training data to fit and test data to evaluate. Evaluate the SVM model that you have created using the metrics of precision, recall, F1-score, and AUROC (area under receiver operating characteristics), and tabulate the results below.[¶](#X65cdb7b63a325fe8454fc2dcfbb8c76b742019e)

In [39]:

short\_titles = ["SVM\_lin-a", "SVM\_lin-b", "SVM\_rbf-a", "SVM\_rbf-b", "SVM\_poly-a", "SVM\_poly-b"]  
titles = ["SVM Lin - age, pregnant, bmi ", "SVM Lin - skin\_thickness, diab\_pedigree, dia\_BP",  
 "SVM rbf - age, pregnant, bmi ", "SVM rbf - skin\_thickness, diab\_pedigree, dia\_BP",  
 "SVM poly - age, pregnant, bmi ", "SVM poly - skin\_thickness, diab\_pedigree, dia\_BP"]  
feature\_sets = [f\_a, f\_b,  
 f\_a, f\_b,  
 f\_a, f\_b]  
  
svm\_clf\_lin = svm.SVC(kernel='linear', probability=True)  
svm\_clf\_rbf = svm.SVC(kernel='rbf', probability=True)  
svm\_clf\_poly = svm.SVC(kernel='poly', probability=True)  
  
classifiers = [svm\_clf\_lin, svm\_clf\_lin, svm\_clf\_rbf, svm\_clf\_rbf, svm\_clf\_poly, svm\_clf\_poly]  
test\_model(classifiers, feature\_sets, short\_titles, titles)

/home/sj/.local/lib/python3.8/site-packages/sklearn/metrics/\_classification.py:846: RuntimeWarning: invalid value encountered in double\_scalars  
 mcc = cov\_ytyp / np.sqrt(cov\_ytyt \* cov\_ypyp)  
/home/sj/.local/lib/python3.8/site-packages/sklearn/metrics/\_classification.py:846: RuntimeWarning: invalid value encountered in double\_scalars  
 mcc = cov\_ytyp / np.sqrt(cov\_ytyt \* cov\_ypyp)  
/home/sj/.local/lib/python3.8/site-packages/sklearn/metrics/\_classification.py:846: RuntimeWarning: invalid value encountered in double\_scalars  
 mcc = cov\_ytyp / np.sqrt(cov\_ytyt \* cov\_ypyp)

![](data:image/png;base64;base64,)

![](data:image/png;base64;base64,)

### 6.5. [1 mark] According to your analysis, answer the following with proper reasoning.[¶](#X75b1ddaa70c19a08ffd42786795a0e2026e53d2)

#### 1. Which SVM classification model kernel performed the best in the classification task?[¶](#X164ffef0b131ca75ee61259de5166ace74d17c3)

rbf kernel performed marginally better than linear kernel when considering recall and F1\_score, while linear has better precision.

#### 2. Did any SVM model perform as bad as the dummy classifier?[¶](#Xb33e2885d20ad35d04c60110ca8baa5a7b5f021)

All kernels using the feature set skin\_thickness, diab\_pedigree, dia\_BP performed similar to the DummyClassifier even though when looking at the ROC it becomes clear that the this feature set performs better on average across all threshold values than the DummyClassifier.

The reason for precision=undefined & recall=0 when using features skin\_thickness, diab\_pedigree, dia\_BP is most likely because $\theta^Tf\lt0$ and thus the predicitons are always $\hat{y} = 0$, which is also evident from the confusion matrices.

#### 3. Which featureset (full, (a), (b)) performed the best in the classification?[¶](#Xd11d3f1ee00dfda1773c47f353f7d65e4a6894b)

The full featureset performed the best.

In [ ]: