1)

1.1) D

1.2) D

2)

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn import svm
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

Q1 Support Vector Machine

Linear SVM

In [2]:
```
# 1) Load data.

X_and_Y = np.load('arrhythmia.npy')    # Load data from file.
np.random.shuffle(X_and_Y)    # Shuffle the data.
X = X_and_Y[:, :279]    # First column to second last column: Features (numerica
Y = X_and_Y[:, 279]   # Last column: Labels (0 or 1)
print(X.shape, Y.shape)       # Check the shapes.
```

```
(452, 279) (452,)
```

In [3]:
```
# 2) Split the dataset into 2 parts:
#    (a) Training set + Validation set  (80% of all data points)
#    (b) Test set                       (20% of all data points)

threshold = round(len(Y) * 0.8)

X_train_val = X[:threshold, :] # Get features from train + val set.
X_test      = X[threshold+1:, :] # Get features from test set.
Y_train_val = Y[:threshold] # Get labels from train + val set.
Y_test      = Y[threshold+1:] # Get labels from test set.
print(X_train_val.shape, X_test.shape, Y_train_val.shape, Y_test.shape)
```

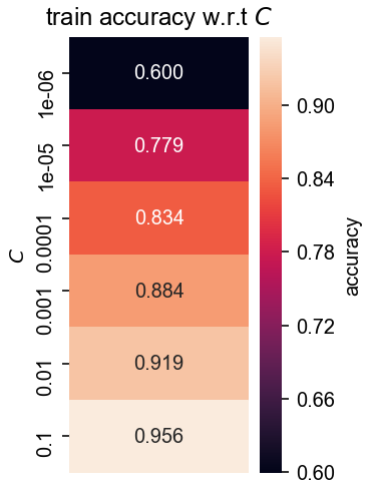
```
(362, 279) (89, 279) (362,) (89,)
```

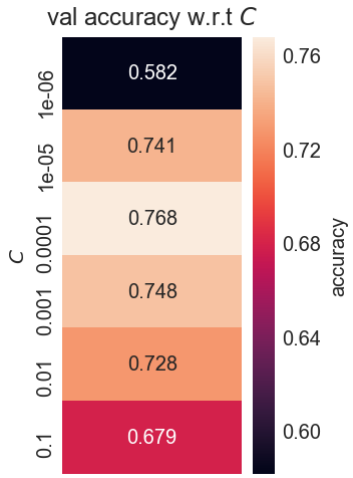
In [4]:
```
# 3) Consider linear kernel. Perform grid search for best C
#    with 3-fold cross-validation. You can use svm.SVC() for SVM
#    classifier and use GridSearchCV() to perform such grid search.
#    For more details, please refer to the sklearn documents:
#         http://scikit-learn.org/stable/modules/svm.html
# http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridS

classifier = svm.SVC(kernel="linear")
C_list     = [10**-6, 10**-5, 10**-4, 10**-3, 10**-2, 10**-1] # Different C to t
parameters = {'C':C_list}

clf = GridSearchCV(classifier, parameters, return_train_score=True)
clf.fit(X, Y)
```

```
C:\Users\Poker\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:19
78: FutureWarning: The default value of cv will change from 3 to 5 in version
0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

Out[4]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
             estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='auto_deprecated', kernel='linear',
                           max_iter=-1, probability=False, random_state=None,
                           shrinking=True, tol=0.001, verbose=False),
             iid='warn', n_jobs=None,
             param_grid={'C': [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring=None, verbose=0)
```

In [5]:
```
# 4) Draw heatmaps for result of grid search and find
#    best C for validation set.

def draw_heatmap_linear(acc, acc_desc, C_list):
    plt.figure(figsize = (2,4))
    ax = sns.heatmap(acc, annot=True, fmt='.3f', yticklabels=C_list, xticklabels
    ax.collections[0].colorbar.set_label("accuracy")
    ax.set(ylabel='$C$')
    plt.title(acc_desc + ' w.r.t $C$')
    sns.set_style("whitegrid", {'axes.grid' : False})
    plt.show()

#
# You can use the draw_heatmap_linear() to draw a heatmap to visualize
# the accuracy w.r.t. C and gamma. Some demo code is given below as hint:
#
# demo_acc        = np.array([[0.8],
#                             [0.7]])
# demo_C_list     = [0.1, 1]
# draw_heatmap_linear(demo_acc, 'demo accuracy', demo_C_list)
#

train_acc = clf.cv_results_["mean_train_score"].reshape(6,1)
draw_heatmap_linear(train_acc, 'train accuracy', C_list)

val_acc = clf.cv_results_["mean_test_score"].reshape(6,1)
draw_heatmap_linear(val_acc, 'val accuracy', C_list)
```





Best C: 0.0001

In [6]:
```
# 5) Use the best C to calculate the test accuracy.
from sklearn.metrics import accuracy_score
lin_svc = svm.SVC(C=0.0001, kernel="linear")
lin_svc.fit(X_train_val, Y_train_val)
predictions = lin_svc.predict(X_test)

test_acc = accuracy_score(Y_test, predictions)
print(test_acc)
```

```
0.7415730337078652
```

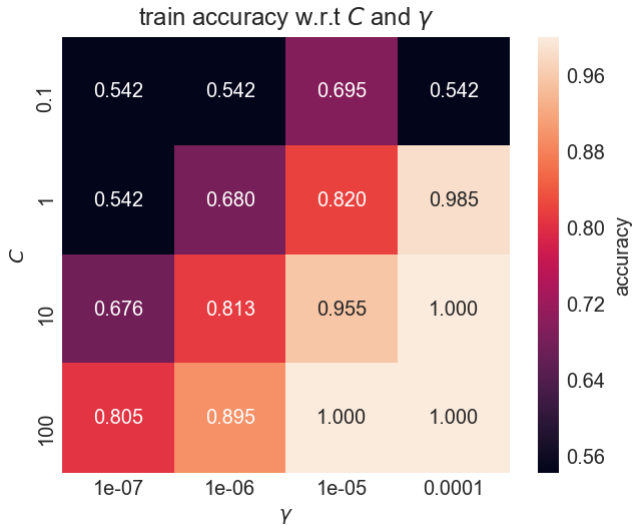
SVM with RBF Kernel

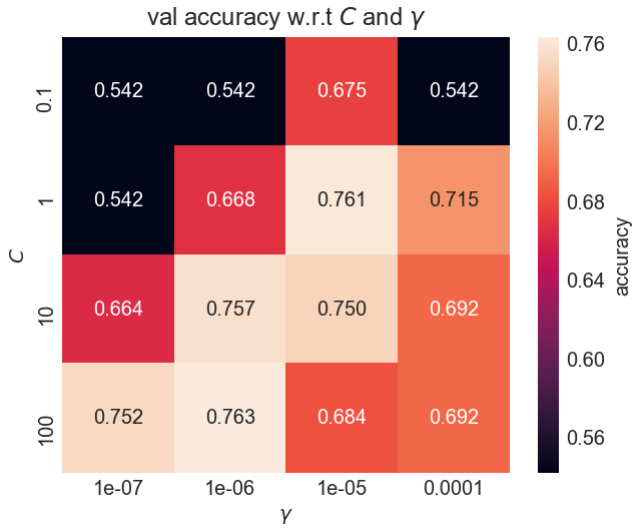
In [7]:
```
# 1) Consider RBF kernel. Perform grid search for best C and gamma
#    with 3-fold cross-validation. You can use svm.SVC() for SVM
#    classifier and use GridSearchCV() to perform such grid search.
#    For more details, please refer to the sklearn documents:
#         http://scikit-learn.org/stable/modules/svm.html
# http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridS
classifier = svm.SVC()
C_list     = [0.1, 1, 10, 100] # Different C to try.
gamma_list = [10**-7, 10**-6, 10**-5, 10**-4] # Different gamma to try.
parameters = {'gamma':gamma_list,'C':C_list}

clf = GridSearchCV(classifier, parameters, return_train_score=True)
clf.fit(X, Y)
```

```
C:\Users\Poker\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:19
78: FutureWarning: The default value of cv will change from 3 to 5 in version
0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

Out[7]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
             estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='auto_deprecated', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=Tru
e,
                           tol=0.001, verbose=False),
             iid='warn', n_jobs=None,
             param_grid={'C': [0.1, 1, 10, 100],
                         'gamma': [1e-07, 1e-06, 1e-05, 0.0001]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring=None, verbose=0)
```

In [8]:
```
# 2) Draw heatmaps for result of grid search and find
#    best C and gamma for validation set.

def draw_heatmap_RBF(acc, acc_desc, gamma_list, C_list):
    plt.figure(figsize = (5,4))
    ax = sns.heatmap(acc, annot=True, fmt='.3f',
                     xticklabels=gamma_list, yticklabels=C_list)
    ax.collections[0].colorbar.set_label("accuracy")
    ax.set(xlabel = '$\gamma$', ylabel='$C$')
    plt.title(acc_desc + ' w.r.t $C$ and $\gamma$')
    sns.set_style("whitegrid", {'axes.grid' : False})
    plt.show()

#
# You can use the draw_heatmap_RBF() to draw a heatmap to visualize
# the accuracy w.r.t. C and gamma. Some demo code is given below as hint:
#
# demo_acc        = np.array([[0.8, 0.7],
#                             [0.7, 0.9]])
# demo_C_list     = [0.1, 1]
# demo_gamma_list = [0.01, 0.1]
# draw_heatmap_RBF(demo_acc, 'demo accuracy', demo_gamma_list, demo_C_list)
#

train_acc = clf.cv_results_["mean_train_score"].reshape(4,4)
draw_heatmap_RBF(train_acc, 'train accuracy', gamma_list, C_list)

val_acc   = clf.cv_results_["mean_test_score"].reshape(4,4)
draw_heatmap_RBF(val_acc, 'val accuracy',gamma_list, C_list)
```





Best C: 100

Best Gamma: 1e-06

In [29]:
```
# 3) Use the best C and gamma to calculate the test accuracy.

c = 1
g = 1**-6
rbf_svc = svm.SVC(C=c, gamma=g)
rbf_svc.fit(X_train_val, Y_train_val)
predictions = rbf_svc.predict(X_test)

test_acc = accuracy_score(Y_test, predictions)
print(test_acc)
```

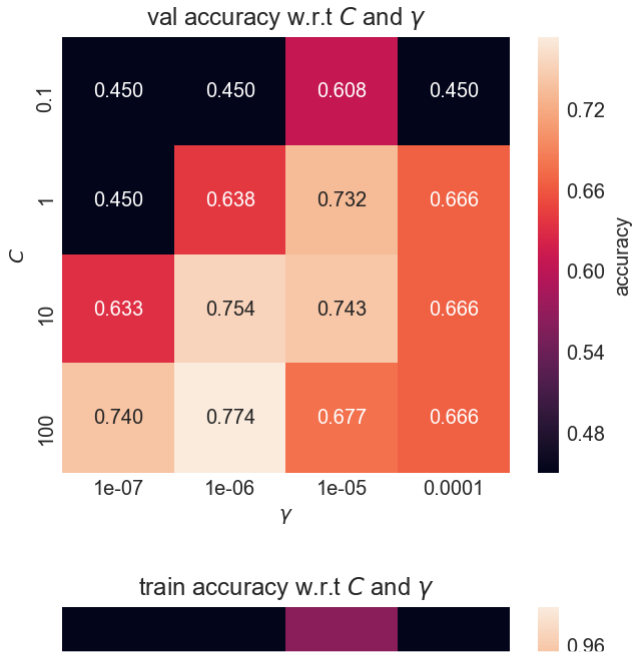
```
0.6067415730337079
```

Re-implementation of Cross-validation and Grid Search

In [10]:
```
# 1) Implement a simple cross-validation.
def simple_cross_validation(X_train_val, Y_train_val, C, gamma, fold):
    """
    A simple cross-validation function.
    We assume the SVM with the RBF kernel.

    X_train_val: Features for train and val set.
                 Shape: (num of data points, num of features)
    Y_train_val: Labels for train and val set.
                 Shape: (num of data points,)
    C:           Parameter C for SVM.
    gamma:       Parameter gamma for SVM.
    fold:        The number of folds to do the cross-validation.

    Return the average accuracy on validation set.
    """

    X_train_val = np.array_split(X_train_val, fold, 0)
    Y_train_val = np.array_split(Y_train_val, fold, 0)

    val_acc_list = []
    train_acc_list = []
    for i in range(fold):

        #split into train and val sets
        X_train = X_train_val.copy()
        Y_train = Y_train_val.copy()
        X_val = X_train.pop(i)
        Y_val = Y_train.pop(i)
        X_train = np.concatenate(X_train)
        Y_train = np.concatenate(Y_train)


        # get accuracies
        svc = svm.SVC(C=C, gamma=gamma)
        svc.fit(X_train, Y_train)

        train_pred = svc.predict(X_train)
        train_acc = accuracy_score(Y_train, train_pred)
        val_pred = svc.predict(X_val)
        val_acc = accuracy_score(Y_val, val_pred)

        val_acc_list.append(val_acc)
        train_acc_list.append(train_acc)

    return sum(val_acc_list) / len(val_acc_list), \
           sum(train_acc_list) / len(train_acc_list)
```

In [11]:
```
# 2) Implement the grid search function.
def simple_GridSearchCV_fit(X_train_val, Y_train_val, C_list, gamma_list, fold):
    """
    A simple grid search function for C and gamma with cross-validation.
    We assume the SVM with the RBF kernel.

    X_train_val: Features for train and val set.
                 Shape: (num of data points, num of features)
    Y_train_val: Labels for train and val set.
                 Shape: (num of data points,)
    C_list:      The list of C values to try.
    gamma_list:  The list of gamma values to try.
    fold:        The number of folds to do the cross-validation.

    Return the val and train accuracy matrix of cross-validation.
    All combinations of C and gamma are
    included in the matrix. Shape: (len(C_list), len(gamma_list))
    """
    val_matrix = np.zeros((len(C_list), len(gamma_list)))
    train_matrix = np.zeros((len(C_list), len(gamma_list)))
    for i in range(len(C_list)):
        for j in range(len(gamma_list)):
            c = C_list[i]
            gamma = gamma_list[j]
            val_matrix[i][j], train_matrix[i][j] = simple_cross_validation(X_tra


    val_acc_matrix   = val_matrix
    train_acc_matrix = train_matrix
    return val_acc_matrix, train_acc_matrix
```

In [12]:
```
# 3) Perform grid search with 3-fold cross-validation.
#    Draw heatmaps for result of grid search and find
#    best C and gamma for validation set.
val_acc_matrix, train_acc_matrix = \
    simple_GridSearchCV_fit(X_train_val, Y_train_val, C_list, gamma_list, 3)

draw_heatmap_RBF(val_acc_matrix, 'val accuracy', gamma_list, C_list)
draw_heatmap_RBF(train_acc_matrix, 'train accuracy', gamma_list, C_list)
```



Best C: 100

Best gamma: 1e-06

In [ ]:
```
```