Lab Assignment 06

The objective of this lab assignment is to build and evaluate classification models to predict customer churn given information from customers of a telephone company (`data_lab_06.csv`).

Instructions:

Complete each task and question by filling in the blanks (`...`) with one or more lines of code or text. Each task and question is worth **0.5 points** (out of **10 points**).

Submission:

This assignment is due **Monday, November 18, at 11:59PM (Central Time)**.

This assignment must be submitted on Gradescope as a **PDF file** containing the completed code for each task and the corresponding output. Late submissions will be accepted within **0-12** hours after the deadline with a **0.5-point (5%) penalty** and within **12-24** hours after the deadline with a **2-point (20%) penalty**. No late submissions will be accepted more than 24 hours after the deadline.

This assignment is individual. Offering or receiving any kind of unauthorized or unacknowledged assistance is a violation of the University's academic integrity policies, will result in a grade of zero for the assignment, and will be subject to disciplinary action.

Part 1: Decision Trees

In [1]:
```
# Load libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn import metrics
from sklearn import tree
from sklearn.externals.six import StringIO
from sklearn import metrics
import pydot
```

```
C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\externals\six.py:31: Deprecationwarning: The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for Python 2.7. Please rely on the official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

In [2]:
```
# Load dataset and display the first five rows
data = pd.read_csv('data_lab_06.csv')
data.head()
```

Out[2]:

	Account length	International plan	Voice mail plan	Number voice mail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	mi
0	128	0	1	25	265.1	110	45.07	197.4	99	16.78	
1	107	0	1	26	161.6	123	27.47	195.5	103	16.62	
2	137	0	0	0	243.4	114	41.38	121.2	110	10.30	
3	84	1	0	0	299.4	71	50.90	61.9	88	5.26	
4	75	1	0	0	166.7	113	28.34	148.3	122	12.61	

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 18 columns):
Account length                3333 non-null int64
International plan            3333 non-null int64
Voice mail plan               3333 non-null int64
Number voice mail messages    3333 non-null int64
Total day minutes             3333 non-null float64
Total day calls               3333 non-null int64
Total day charge              3333 non-null float64
Total eve minutes             3333 non-null float64
Total eve calls               3333 non-null int64
Total eve charge              3333 non-null float64
Total night minutes           3333 non-null float64
Total night calls             3333 non-null int64
Total night charge            3333 non-null float64
Total intl minutes            3333 non-null float64
Total intl calls              3333 non-null int64
Total intl charge             3333 non-null float64
Customer service calls        3333 non-null int64
Churn                         3333 non-null int64
dtypes: float64(8), int64(10)
memory usage: 468.8 KB
```

Task 01 (of 14): Partition the dataset into training set and test set. *Hint:* Use 75% of the data for training and 25% for testing and set parameter random_state to 0.

In [5]:
```
x_train, x_test, y_train, y_test = train_test_split(data.iloc[:,:17],data.iloc
[:,17:],test_size = 0.25, random_state=0)
```

In [6]:
```
# Show the dimensionality of the training set and the test set
# The training set should have 2499 observations and the test set should have
 834 observations
print(x_train.shape)
print(x_test.shape)
```

```
(2499, 17)
(834, 17)
```

In [7]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2499 entries, 427 to 2732
Data columns (total 17 columns):
Account length                2499 non-null int64
International plan            2499 non-null int64
Voice mail plan               2499 non-null int64
Number voice mail messages    2499 non-null int64
Total day minutes             2499 non-null float64
Total day calls               2499 non-null int64
Total day charge              2499 non-null float64
Total eve minutes             2499 non-null float64
Total eve calls               2499 non-null int64
Total eve charge              2499 non-null float64
Total night minutes           2499 non-null float64
Total night calls             2499 non-null int64
Total night charge            2499 non-null float64
Total intl minutes            2499 non-null float64
Total intl calls              2499 non-null int64
Total intl charge             2499 non-null float64
Customer service calls        2499 non-null int64
dtypes: float64(8), int64(9)
memory usage: 351.4 KB
```

In [8]:
```
columns = x_train.columns
columns
```

Out[8]:
```
Index(['Account length', 'International plan', 'Voice mail plan',
       'Number voice mail messages', 'Total day minutes', 'Total day calls',
       'Total day charge', 'Total eve minutes', 'Total eve calls',
       'Total eve charge', 'Total night minutes', 'Total night calls',
       'Total night charge', 'Total intl minutes', 'Total intl calls',
       'Total intl charge', 'Customer service calls'],
      dtype='object')
```

Task 02 (of 14): Standardize the training set and test set. *Hint:* Compute the mean and standard deviation using only the training set and then apply this transformation on the training set and test set.

In [9]:
```
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Task 03 (of 14): Build a decision tree classifier to classify customers as churnend/non-churned. *Hint:* Use entropy as the split criterion.

In [10]:
```
classifier = DecisionTreeClassifier(criterion = "entropy")
x = x_train_scaled
y = y_train
classifier.fit(x,y)
```

Out[10]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=Non
e,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [11]:
```
# Show the structure of the decision tree classifier
print(classifier.tree_.__getstate__()['nodes'])
len(classifier.tree_.__getstate__()['nodes'])
```

```
[(  1, 240,  4,  1.37790751e+00, 0.60293799, 2499, 2.499e+03)
 (  2, 201, 16,  1.47932547e+00, 0.4991475 , 2278, 2.278e+03)
 (  3, 188,  1,  1.39084876e+00, 0.37934172, 2105, 2.105e+03)
 (  4, 137,  6,  7.91448623e-01, 0.26832186, 1921, 1.921e+03)
 (  5,   6, 12, -1.40246254e+00, 0.1779243 , 1681, 1.681e+03)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,  124, 1.240e+02)
 (  7,   8, 10, -1.40125614e+00, 0.18885385, 1557, 1.557e+03)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 (  9,  26,  7, -6.43222064e-01, 0.18567934, 1556, 1.556e+03)
 ( 10,  25, 13,  3.73752005e-02, 0.09482908,  411, 4.110e+02)
 ( 11,  22, 15, -2.16688029e-03, 0.16417121,  207, 2.070e+02)
 ( 12,  17,  3,  1.93759680e+00, 0.11102003,  203, 2.030e+02)
 ( 13,  16,  0, -1.88109082e+00, 0.04741446,  190, 1.900e+02)
 ( 14,  15,  5, -1.34136140e+00, 0.72192809,    5, 5.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,  185, 1.850e+02)
 ( 18,  19,  0,  9.24853444e-01, 0.61938219,   13, 1.300e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
 ( 20,  21, 16, -4.31019366e-02, 1.        ,    4, 4.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 ( 23,  24, 12,  7.08070576e-01, 1.        ,    4, 4.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,  204, 2.040e+02)
 ( 27,  92, 11,  7.33808935e-01, 0.21436617, 1145, 1.145e+03)
 ( 28,  29, 13, -7.10542113e-01, 0.17017729,  870, 8.700e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,  197, 1.970e+02)
 ( 30,  45,  0, -1.02322608e-01, 0.20770499,  673, 6.730e+02)
 ( 31,  32, 10,  6.39307916e-01, 0.11670201,  318, 3.180e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,  214, 2.140e+02)
 ( 33,  34, 12,  6.53279960e-01, 0.27817101,  104, 1.040e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 ( 35,  40,  8,  8.06315780e-01, 0.23692475,  103, 1.030e+02)
 ( 36,  37,  5,  2.11814457e+00, 0.09227725,   85, 8.500e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   83, 8.300e+01)
 ( 38,  39,  8, -7.11721629e-01, 1.        ,    2, 2.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 ( 41,  44,  5, -5.44928372e-01, 0.65002242,   18, 1.800e+01)
 ( 42,  43,  6, -9.98059690e-01, 0.97095059,    5, 5.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   13, 1.300e+01)
 ( 46,  53,  0, -2.11031316e-03, 0.27735376,  355, 3.550e+02)
 ( 47,  52,  9,  1.43753242e+00, 0.73828487,   24, 2.400e+01)
 ( 48,  49,  8,  1.59283437e-01, 0.5746357 ,   22, 2.200e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   15, 1.500e+01)
 ( 50,  51, 11, -4.60047260e-01, 0.98522814,    7, 7.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 ( 54,  75,  8, -6.37064070e-01, 0.22484398,  331, 3.310e+02)
 ( 55,  64,  0,  1.98314279e-01, 0.40707681,   86, 8.600e+01)
 ( 56,  59, 11, -5.10849640e-01, 0.83664074,   15, 1.500e+01)
 ( 57,  58,  5, -5.44928387e-01, 0.81127812,    4, 4.000e+00)
```

```
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
( 60,  63,  9, -5.43136686e-01, 0.43949699,   11, 1.100e+01)
( 61,  62, 15,  4.66124743e-01, 1.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
( 65,  70,  5,  1.02304912e+00, 0.25253077,   71, 7.100e+01)
( 66,  67,  6,  6.59875095e-01, 0.12741851,   57, 5.700e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   54, 5.400e+01)
( 68,  69,  6,  7.01025665e-01, 0.91829583,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( 71,  74,  9,  2.71891959e-01, 0.59167278,   14, 1.400e+01)
( 72,  73,  9, -8.67666295e-02, 0.97095059,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
( 76,  91,  0,  3.01678514e+00, 0.14372617,  245, 2.450e+02)
( 77,  78,  7, -6.26610994e-01, 0.12068101,  244, 2.440e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( 79,  90, 12, -6.55724183e-02, 0.0959704 ,  243, 2.430e+02)
( 80,  89, 12, -8.31054039e-02, 0.19143325,  102, 1.020e+02)
( 81,  82,  0,  6.99375778e-01, 0.14032727,  101, 1.010e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   57, 5.700e+01)
( 83,  88,  0,  7.49481887e-01, 0.26676499,   44, 4.400e+01)
( 84,  85, 15, -2.52801836e-01, 0.97095059,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( 86,  87, 12, -3.72399658e-01, 0.91829583,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   39, 3.900e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,  141, 1.410e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( 93, 118, 12,  7.38753289e-01, 0.33462053,  275, 2.750e+02)
( 94, 111,  0,  1.28812295e+00, 0.2085566 ,  213, 2.130e+02)
( 95, 100,  3,  1.42354321e+00, 0.14431028,  195, 1.950e+02)
( 96,  97,  8,  1.25426126e+00, 0.05390791,  163, 1.630e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,  146, 1.460e+02)
( 98,  99,  4, -1.03005683e+00, 0.32275696,   17, 1.700e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   16, 1.600e+01)
(101, 110, 12, -4.46914852e-01, 0.44886449,   32, 3.200e+01)
(102, 103, 13, -5.16625764e-02, 0.72192809,   15, 1.500e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    7, 7.000e+00)
(104, 105, 10, -9.28839415e-01, 0.954434  ,    8, 8.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
(106, 109, 15,  1.71270394e+00, 1.        ,    6, 6.000e+00)
(107, 108,  3,  2.45165038e+00, 0.81127812,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   17, 1.700e+01)
(112, 117, 12, -1.95483305e-02, 0.65002242,   18, 1.800e+01)
(113, 116, 11,  1.26723403e+00, 0.98522814,    7, 7.000e+00)
(114, 115,  0,  1.42591488e+00, 0.72192809,    5, 5.000e+00)
```

```
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   11, 1.100e+01)
(119, 136, 11,  1.49584478e+00, 0.6373875 ,   62, 6.200e+01)
(120, 123,  7, -4.18483824e-01, 0.81127812,   40, 4.000e+01)
(121, 122, 12,  9.18466389e-01, 0.72192809,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
(124, 135,  9,  1.98931485e+00, 0.66096234,   35, 3.500e+01)
(125, 128, 12,  8.50526065e-01, 0.53283506,   33, 3.300e+01)
(126, 127,  8,  6.07228905e-01, 0.91829583,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(129, 134,  5, -1.71600401e-01, 0.35335934,   30, 3.000e+01)
(130, 131,  5, -4.95151341e-01, 0.65002242,   12, 1.200e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
(132, 133, 14,  4.20300819e-01, 1.        ,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   18, 1.800e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   22, 2.200e+01)
(138, 179,  7,  1.13709563e+00, 0.67825063,  240, 2.400e+02)
(139, 174,  9,  8.44366252e-01, 0.41666476,  202, 2.020e+02)
(140, 173,  5,  4.50612903e-01, 0.34918437,  183, 1.830e+02)
(141, 142,  4,  7.96166897e-01, 0.44412605,  130, 1.300e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(143, 144,  0, -2.01888275e+00, 0.42048596,  129, 1.290e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(145, 172,  0,  9.74959582e-01, 0.39553781,  128, 1.280e+02)
(146, 169,  5,  4.00835827e-01, 0.45969421,  103, 1.030e+02)
(147, 148, 13, -1.01327056e+00, 0.40502013,   99, 9.900e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   18, 1.800e+01)
(149, 150,  0, -1.90614390e+00, 0.46506984,   81, 8.100e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(151, 168, 13,  5.00371635e-01, 0.42806963,   80, 8.000e+01)
(152, 165, 15,  3.93572524e-01, 0.59167278,   49, 4.900e+01)
(153, 154, 10, -1.39632481e+00, 0.49596907,   46, 4.600e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(155, 156, 12, -2.38710649e-01, 0.43275016,   45, 4.500e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   17, 1.700e+01)
(157, 158, 12, -1.99261434e-01, 0.59167278,   28, 2.800e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(159, 160,  7, -4.13598210e-01, 0.50325833,   27, 2.700e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   15, 1.500e+01)
(161, 164, 10,  1.08706820e+00, 0.81127812,   12, 1.200e+01)
(162, 163, 14, -8.07439417e-01, 0.46899559,   10, 1.000e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
(166, 167,  4,  1.08335525e+00, 0.91829583,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   31, 3.100e+01)
(170, 171, 14, -8.07439417e-01, 1.        ,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
```

```
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   25, 2.500e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   53, 5.300e+01)
(175, 176, 10,  6.82655824e-02, 0.83147439,   19, 1.900e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   10, 1.000e+01)
(177, 178,  3,  2.48563558e-01, 0.99107606,    9, 9.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
(180, 187,  3, -1.92053802e-01, 0.89974376,   38, 3.800e+01)
(181, 186,  7,  1.31590915e+00, 0.56650951,   30, 3.000e+01)
(182, 183,  7,  1.19572300e+00, 0.99107606,    9, 9.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
(184, 185, 10,  4.86438192e-01, 0.72192809,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   21, 2.100e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
(189, 190, 14, -8.07439417e-01, 0.94605843,  184, 1.840e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   36, 3.600e+01)
(191, 200, 13,  1.01679072e+00, 0.74044825,  148, 1.480e+02)
(192, 199,  9,  2.40889943e+00, 0.24678396,  122, 1.220e+02)
(193, 196,  6,  1.32099146e+00, 0.16866093,  120, 1.200e+02)
(194, 195, 10, -2.31255686e+00, 0.07099895,  117, 1.170e+02)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,  116, 1.160e+02)
(197, 198,  9, -1.55739438e-01, 0.91829583,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   26, 2.600e+01)
(202, 213,  4, -3.67314354e-01, 0.99302328,  173, 1.730e+02)
(203, 208,  9,  6.19055063e-01, 0.37123233,   70, 7.000e+01)
(204, 205,  8,  2.29946733e+00, 0.12741851,   57, 5.700e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   55, 5.500e+01)
(206, 207,  1,  1.39084876e+00, 1.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(209, 210,  4, -8.38597924e-01, 0.89049164,   13, 1.300e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    7, 7.000e+00)
(211, 212, 13,  2.51065865e-01, 0.91829583,    6, 6.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
(214, 219,  7, -9.00205344e-01, 0.87034605,  103, 1.030e+02)
(215, 216,  0,  8.12114596e-01, 0.35335934,   15, 1.500e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   13, 1.300e+01)
(217, 218, 15,  2.48468079e-01, 1.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(220, 225,  4, -1.21547356e-01, 0.68403844,   88, 8.800e+01)
(221, 224,  9,  2.02919155e-01, 0.99277445,   20, 2.000e+01)
(222, 223,  6, -3.42357844e-01, 0.46899559,   10, 1.000e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   10, 1.000e+01)
(226, 235,  1,  1.39084876e+00, 0.47825016,   68, 6.800e+01)
(227, 228, 11,  4.03593391e-01, 0.21357982,   59, 5.900e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   42, 4.200e+01)
```

```
(229, 234, 14, -3.98192666e-01, 0.52255937,   17, 1.700e+01)
(230, 231,  7,  1.55087259e-01, 0.97095059,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
(232, 233,  7,  8.49821568e-01, 0.91829583,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   12, 1.200e+01)
(236, 239,  9,  7.24813387e-01, 0.99107606,    9, 9.000e+00)
(237, 238, 10,  1.44207310e-01, 0.72192809,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
(241, 276,  3, -1.18617579e-01, 0.99667435,  221, 2.210e+02)
(242, 261,  9, -2.58406526e-02, 0.92594006,  170, 1.700e+02)
(243, 260,  6,  2.41256452e+00, 0.97663491,   78, 7.800e+01)
(244, 249, 10, -1.28985655e-01, 0.92752659,   70, 7.000e+01)
(245, 246,  4,  1.90441960e+00, 0.36205125,   29, 2.900e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   24, 2.400e+01)
(247, 248,  8, -5.62406461e-01, 0.97095059,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
(250, 251,  9, -1.21562153e+00, 0.99613448,   41, 4.100e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
(252, 259,  4,  1.72308588e+00, 0.91829583,   33, 3.300e+01)
(253, 258, 12,  1.57157004e+00, 0.99836367,   21, 2.100e+01)
(254, 255,  9, -4.28182021e-01, 0.89603823,   16, 1.600e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
(256, 257,  8, -3.63319598e-01, 0.86312057,    7, 7.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   12, 1.200e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
(262, 269, 12, -8.72089744e-01, 0.55862937,   92, 9.200e+01)
(263, 268,  4,  1.78567821e+00, 0.99750255,   17, 1.700e+01)
(264, 265, 14,  1.10540837e-02, 0.68403844,   11, 1.100e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
(266, 267,  8,  3.83256167e-01, 0.91829583,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    6, 6.000e+00)
(270, 275,  6,  1.48451084e+00, 0.24229219,   75, 7.500e+01)
(271, 274, 14, -8.07439417e-01, 0.81127812,   12, 1.200e+01)
(272, 273, 12,  3.46452728e-01, 0.97095059,    5, 5.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    7, 7.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   63, 6.300e+01)
(277, 280,  1,  1.39084876e+00, 0.52255937,   51, 5.100e+01)
(278, 279,  4,  2.53954768e+00, 0.15935006,   43, 4.300e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,   42, 4.200e+01)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
(281, 284,  0, -3.96899208e-02, 0.954434  ,    8, 8.000e+00)
(282, 283, 14, -8.07439417e-01, 0.81127812,    4, 4.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    3, 3.000e+00)
( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)]
```

Out[11]: 285

In [12]:
```
# Plot decision tree
dot_data = StringIO()
tree.export_graphviz(classifier, out_file = dot_data, feature_names = columns)
figure = pydot.graph_from_dot_data(dot_data.getvalue())
figure[0].write_pdf("Lab6ree.pdf")
```

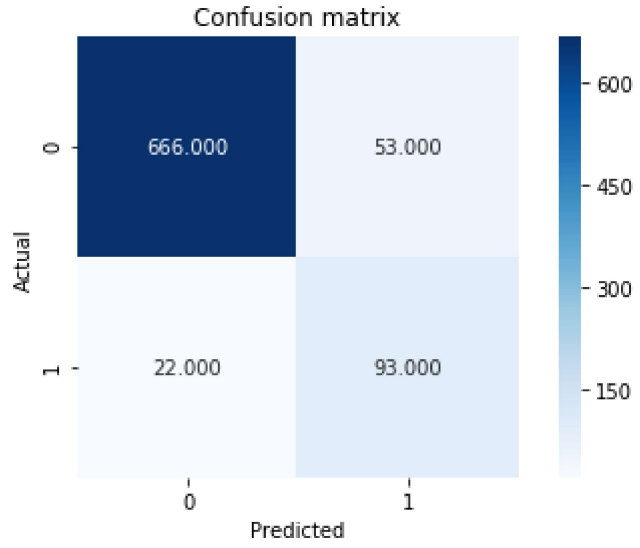
Question 01 (of 06): How many nodes are in the tree? Which variable was selected to split the root node of the tree? What can you conclude from observing the structure of the tree?

Answer: There are 283 nodes in the tree. The variable "Total day minutes" was selected to split the root node. The tree is left-skewed and has large number of nodes which might lead to overfitting. So, it's better to prune the tree.

Task 04 (of 14): Predict the class labels for the test set using the decision tree classifier and plot the corresponding confusion matrix.

In [38]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [39]:
```
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



Task 05 (of 14): Compute evaluation metrics for the decision tree classifier.

In [14]:
```
# Compute evaluation metrics

accuracy = metrics.accuracy_score(y_test, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_test, y_pred, average = None)
recall = metrics.recall_score(y_test, y_pred, average = None)
F1_score = metrics.f1_score(y_test, y_pred, average = None)

print([accuracy, error, precision, recall, F1_score])
```

```
[0.9112709832134293, 0.08872901678657075, array([0.96806967, 0.64137931]), ar
ray([0.92767733, 0.80869565]), array([0.94744318, 0.71538462])]
```

Question 02 (of 06): What can you conclude about the performance of the decision tree classifier?

Answer: From the F1 score values, we can conclude that the performance of the decision tree classifier for class 0 is good but for class 1 it is not that good.

Part 2: k-Nearest Neighbors

Task 06 (of 14): Build a k-nearest neighbors classifier to classify customers as churnend/non-churned.
Hint: Use *k=3* as the number of nearest neighbors.

In [13]:
```
classifier = KNeighborsClassifier(n_neighbors = 3)
classifier.fit(x_train_scaled, y_train)
```

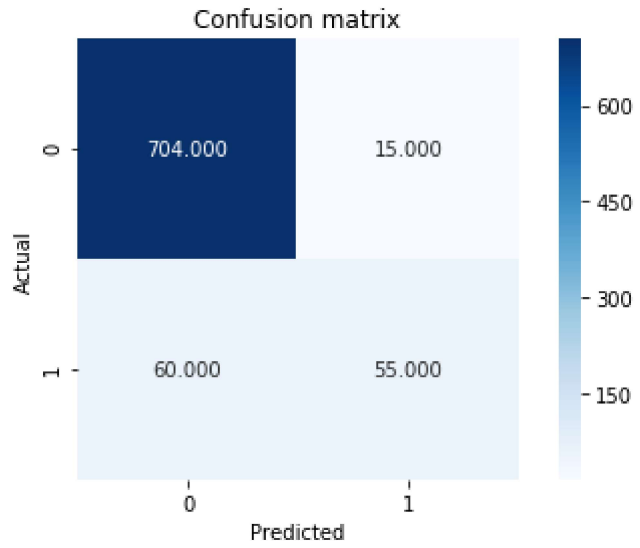
```
C:\Users\kalya\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Plea
se change the shape of y to (n_samples, ), for example using ravel().
```

Out[13]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

Task 07 (of 14): Predict the class labels for the test set using the k-nearest neighbors classifier and plot the corresponding confusion matrix.

In [14]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [15]:
```
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



Task 08 (of 14): Compute evaluation metrics for the k-nearest neighbors classifier.

In [16]:
```
# Compute evaluation metrics

accuracy = metrics.accuracy_score(y_test, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_test, y_pred, average = None)
recall = metrics.recall_score(y_test, y_pred, average = None)
F1_score = metrics.f1_score(y_test, y_pred, average = None)

print([accuracy, error, precision, recall, F1_score])
```

```
[0.9100719424460432, 0.08992805755395683, array([0.92146597, 0.78571429]), ar
ray([0.97913769, 0.47826087]), array([0.94942684, 0.59459459])]
```

Question 03 (of 06): What can you conclude about the performance of the k-nearest neighbors classifier?

Answer: From the F1 score values, we can conclude that the performance of the decision tree classifier for class 0 is good but for class 1 is average.

Part 3: Naive Bayes

Task 09 (of 14): Build a Naive Bayes classifier to classify customers as churnend/non-churned.

In [17]:
```
classifier = GaussianNB()
classifier.fit(x_train_scaled, y_train)
```

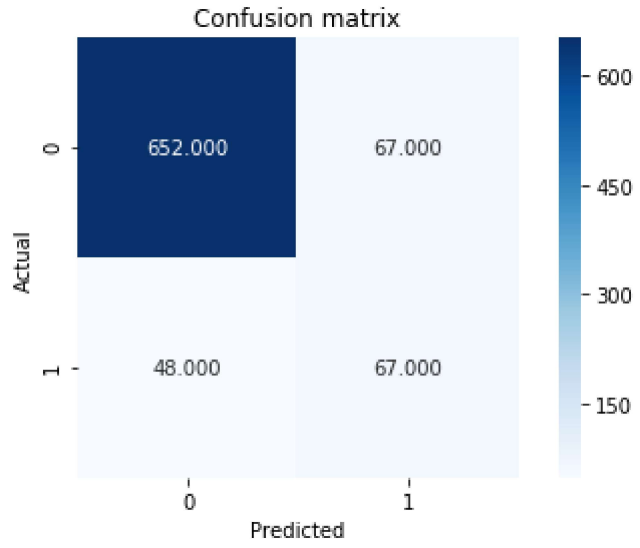
```
C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: D
ataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[17]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Task 10 (of 14): Predict the class labels for the test set using the Naive Bayes classifier and plot the corresponding confusion matrix.

In [18]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [19]:
```
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



Task 11 (of 14): Compute evaluation metrics for the Naive Bayes classifier.

In [20]:
```
# Compute evaluation metrics

accuracy = metrics.accuracy_score(y_test, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_test, y_pred, average = None)
recall = metrics.recall_score(y_test, y_pred, average = None)
F1_score = metrics.f1_score(y_test, y_pred, average = None)

print([accuracy, error, precision, recall, F1_score])
```

```
[0.8621103117505995, 0.1378896882494005, array([0.93142857, 0.5       ]), arr
ay([0.90681502, 0.5826087 ]), array([0.91895701, 0.53815261])]
```

Question 04 (of 06): What can you conclude about the performance of the Naive Bayes classifier?

Answer: From the F1 score values, we can conclude that the performance of the decision tree classifier for class 0 is good but for class 1 is average.

Part 4: Support Vector Machines

Task 12 (of 14): Build an SVM classifier to classify customers as churnend/non-churned. *Hint:* Use *rbf* (radial basis function) as the kernel function.

In [21]:
```
classifier = SVC(kernel = 'rbf')
classifier.fit(x_train_scaled, y_train)
```

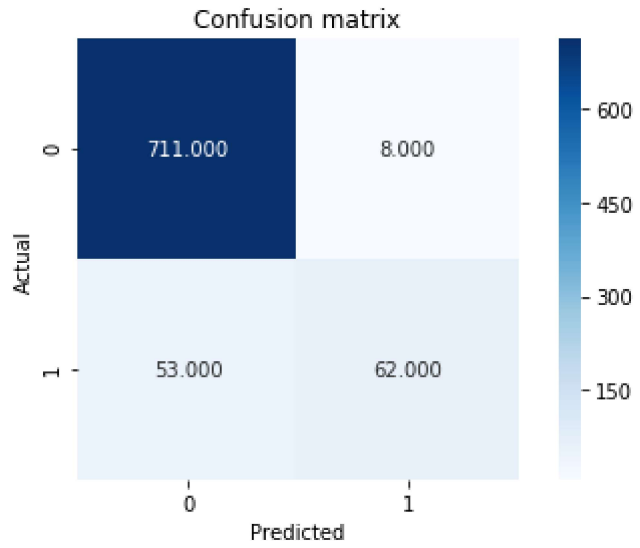
```
C:\Users\kalya\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: D
ataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[21]:
```
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=True, tol=0.001, verbose=False)
```

Task 13 (of 14): Predict the class labels for the test set using the SVM classifier and plot the corresponding confusion matrix.

In [22]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [23]:
```
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.tight_layout()
```



Task 14 (of 14): Compute evaluation metrics for the SVM classifier.

In [24]:
```
# Compute evaluation metrics

accuracy = metrics.accuracy_score(y_test, y_pred)
error = 1 - accuracy
precision = metrics.precision_score(y_test, y_pred, average = None)
recall = metrics.recall_score(y_test, y_pred, average = None)
F1_score = metrics.f1_score(y_test, y_pred, average = None)

print([accuracy, error, precision, recall, F1_score])
```

```
[0.9268585131894485, 0.07314148681055155, array([0.93062827, 0.88571429]), ar
ray([0.98887344, 0.53913043]), array([0.95886716, 0.67027027])]
```

Question 05 (of 06): What can you conclude about the performance of the SVM classifier?

Answer: From the F1 score values, we can conclude that the performance of the decision tree classifier for class 0 is good but for class 1 is above average.

Question 06 (of 06): Which of the classifiers had the best performance?

Answer: Based on the F1 score arrays of the classifiers, the decision tree classifier has the best performance.

In [ ]: