Lab Assignment 04

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_04.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 **Sunday, October 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. To save your Jupyter notebook as a PDF file, go to `File > Export Notebook As > HTML` or `File > Download As > HTML` , open the HTML file and print it as a PDF file. Additionally, this assignment has a single question on Gradescope and **all pages of the PDF file** must be assigned to this question. A **0.5-point (5%) penalty** will be applied to submissions that do not follow these guidelines. For more instructions on how to submit assignments on Gradescope, see this guide.

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 [24]:
```
# 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
```

In [25]:
```
# Load dataset and display the first five rows
data = pd.read_csv('data_lab_04.csv')
data.head()
```

Out[25]:

	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
0	128	0	1	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3
1	107	0	1	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3
2	137	0	0	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5
3	84	1	0	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7
4	75	1	0	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3

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

In [26]:
```
x_train, x_test, y_train, y_test = train_test_split(data.iloc[:,0:17], data.iloc[:,-1], test_size = 0.2
5, random_state = 0)
```

In [27]:
```
# 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)
```

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

In [28]:
```
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. Use entropy as the split criterion and set parameter `random_state` to 0.

In [29]:
```
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train_scaled, y_train)
```

Out[29]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [30]:
```
# Show the structure of the decision tree classifier
print(classifier.tree_.__getstate__()['nodes'])
len(classifier.tree_.__getstate__()['nodes'])
```

```
[(  1, 238,  4,  1.37790751e+00, 0.60293799, 2499, 2.499e+03)
 (  2, 199, 16,  1.47932547e+00, 0.4991475 , 2278, 2.278e+03)
 (  3, 186,  1,  1.39084876e+00, 0.37934172, 2105, 2.105e+03)
 (  4, 135,  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, 13,  1.76009070e-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,  7, -1.54412925e+00, 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,  7, -1.24317545e+00, 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, 12,  6.40130222e-01, 0.11670201,  318, 3.180e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,  214, 2.140e+02)
 ( 33,  34, 10,  6.54101759e-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,  5,  2.59102666e+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)
 ( 41,  44,  5, -5.44928372e-01, 0.65002242,   18, 1.800e+01)
 ( 42,  43,  0, -4.15486038e-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.        ,   13, 1.300e+01)
 ( 46,  53,  0, -2.11031316e-03, 0.27735376,  355, 3.550e+02)
 ( 47,  52,  7,  1.43707240e+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,  4,  8.18777606e-02, 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,  7, -5.43555543e-01, 0.43949699,   11, 1.100e+01)
 ( 61,  62,  7, -6.02182925e-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,  4,  6.59936458e-01, 0.12741851,   57, 5.700e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   54, 5.400e+01)
 ( 68,  69,  0,  8.74747276e-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)
 ( 71,  74,  7,  2.71364868e-01, 0.59167278,   14, 1.400e+01)
 ( 72,  73,  7, -8.62620762e-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,  9, -6.27053618e-01, 0.12068101,  244, 2.440e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 ( 79,  90, 10, -6.68515153e-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, 14, -3.98192666e-01, 0.97095059,    5, 5.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 ( 86,  87, 10, -3.62728354e-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, 116, 10,  7.37933517e-01, 0.33462053,  275, 2.750e+02)
 ( 94, 109,  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,  6, -1.03000554e+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, 102,  8, -1.14461008e-01, 0.44886449,   32, 3.200e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   17, 1.700e+01)
 (103, 108,  5,  1.51950516e-01, 0.72192809,   15, 1.500e+01)
 (104, 107,  8,  1.00540265e+00, 0.954434  ,    8, 8.000e+00)
 (105, 106,  9,  9.76564109e-01, 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.        ,    4, 4.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    7, 7.000e+00)
 (110, 115, 10, -2.04974730e-02, 0.65002242,   18, 1.800e+01)
 (111, 114, 11,  1.26723403e+00, 0.98522814,    7, 7.000e+00)
 (112, 113,  7, -8.52849483e-02, 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)
 (117, 134, 11,  1.49584478e+00, 0.6373875 ,   62, 6.200e+01)
 (118, 121,  7, -4.18483824e-01, 0.81127812,   40, 4.000e+01)
 (119, 120, 16,  3.37504923e-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)
 (122, 133,  7,  1.99012387e+00, 0.66096234,   35, 3.500e+01)
 (123, 126, 10,  8.52339238e-01, 0.53283506,   33, 3.300e+01)
 (124, 125, 15,  3.93572517e-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)
 (127, 132,  5, -1.71600401e-01, 0.35335934,   30, 3.000e+01)
 (128, 129,  5, -4.95151341e-01, 0.65002242,   12, 1.200e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
 (130, 131,  4, -1.53764002e-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)
 (136, 177,  9,  1.13635111e+00, 0.67825063,  240, 2.400e+02)
 (137, 172,  7,  8.43958825e-01, 0.41666476,  202, 2.020e+02)
 (138, 171,  5,  4.50612903e-01, 0.34918437,  183, 1.830e+02)
 (139, 140,  4,  7.96166897e-01, 0.44412605,  130, 1.300e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 (141, 142,  0, -2.01888275e+00, 0.42048596,  129, 1.290e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 (143, 170,  0,  9.74959582e-01, 0.39553781,  128, 1.280e+02)
 (144, 167,  5,  4.00835827e-01, 0.45969421,  103, 1.030e+02)
 (145, 146, 15, -1.01130235e+00, 0.40502013,   99, 9.900e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   18, 1.800e+01)
 (147, 148,  0, -1.90614390e+00, 0.46506984,   81, 8.100e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 (149, 166, 15,  4.99103040e-01, 0.42806963,   80, 8.000e+01)
 (150, 163, 13,  3.93526301e-01, 0.59167278,   49, 4.900e+01)
 (151, 152, 10, -1.39632481e+00, 0.49596907,   46, 4.600e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 (153, 154, 12, -2.38710649e-01, 0.43275016,   45, 4.500e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   17, 1.700e+01)
 (155, 156, 12, -1.99261434e-01, 0.59167278,   28, 2.800e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 (157, 158,  7, -4.13598210e-01, 0.50325833,   27, 2.700e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   15, 1.500e+01)
 (159, 162, 10,  1.08706820e+00, 0.81127812,   12, 1.200e+01)
 (160, 161, 11,  1.03862327e+00, 0.46899559,   10, 1.000e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    1, 1.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 (164, 165, 14,  1.64804107e+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)
 (168, 169,  7, -7.43865728e-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)
 (173, 174, 10,  6.82655824e-02, 0.83147439,   19, 1.900e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   10, 1.000e+01)
 (175, 176,  2,  4.90775019e-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)
 (178, 185,  3, -1.92053802e-01, 0.89974376,   38, 3.800e+01)
 (179, 184,  9,  1.31568044e+00, 0.56650951,   30, 3.000e+01)
 (180, 181,  7,  1.19572300e+00, 0.99107606,    9, 9.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    4, 4.000e+00)
 (182, 183,  5,  1.52081978e+00, 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)
 (187, 188, 14, -8.07439417e-01, 0.94605843,  184, 1.840e+02)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   36, 3.600e+01)
 (189, 198, 15,  1.01356426e+00, 0.74044825,  148, 1.480e+02)
 (190, 197,  7,  2.40833235e+00, 0.24678396,  122, 1.220e+02)
 (191, 194,  6,  1.32099146e+00, 0.16866093,  120, 1.200e+02)
 (192, 193, 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)
 (195, 196,  7, -1.56614915e-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)
 (200, 211,  6, -3.67264792e-01, 0.99302328,  173, 1.730e+02)
 (201, 206,  7,  6.19220585e-01, 0.37123233,   70, 7.000e+01)
 (202, 203,  8,  2.29946733e+00, 0.12741851,   57, 5.700e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   55, 5.500e+01)
 (204, 205,  8,  2.69764102e+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)
 (207, 208,  4, -8.38597924e-01, 0.89049164,   13, 1.300e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    7, 7.000e+00)
 (209, 210, 15,  2.48468079e-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)
 (212, 217,  9, -8.99496198e-01, 0.87034605,  103, 1.030e+02)
 (213, 214,  0,  8.12114596e-01, 0.35335934,   15, 1.500e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   13, 1.300e+01)
 (215, 216,  2,  4.90775019e-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)
 (218, 223,  6, -1.21444274e-01, 0.68403844,   88, 8.800e+01)
 (219, 222,  9,  2.02919155e-01, 0.99277445,   20, 2.000e+01)
 (220, 221,  4, -3.42461511e-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)
 (224, 233,  1,  1.39084876e+00, 0.47825016,   68, 6.800e+01)
 (225, 226, 11,  4.03593391e-01, 0.21357982,   59, 5.900e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   42, 4.200e+01)
 (227, 232, 14, -3.98192666e-01, 0.52255937,   17, 1.700e+01)
 (228, 229,  9,  1.54638190e-01, 0.97095059,    5, 5.000e+00)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    2, 2.000e+00)
 (230, 231, 10, -1.33616322e+00, 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.        ,   12, 1.200e+01)
 (234, 237,  7,  7.25726977e-01, 0.99107606,    9, 9.000e+00)
 (235, 236, 12,  1.42631780e-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)
 (239, 274,  2,  4.90775019e-01, 0.99667435,  221, 2.210e+02)
 (240, 259,  7, -2.56804614e-02, 0.92594006,  170, 1.700e+02)
 (241, 258,  4,  2.41252208e+00, 0.97663491,   78, 7.800e+01)
 (242, 247, 12, -1.29129488e-01, 0.92752659,   70, 7.000e+01)
 (243, 244,  6,  1.90413839e+00, 0.36205125,   29, 2.900e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,   24, 2.400e+01)
 (245, 246,  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)
 (248, 249,  9, -1.21562153e+00, 0.99613448,   41, 4.100e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
 (250, 257,  6,  1.72329241e+00, 0.91829583,   33, 3.300e+01)
 (251, 256, 10,  1.57132000e+00, 0.99836367,   21, 2.100e+01)
 (252, 253,  7, -4.27277938e-01, 0.89603823,   16, 1.600e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    9, 9.000e+00)
 (254, 255,  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)
 (260, 267, 12, -8.72089744e-01, 0.55862937,   92, 9.200e+01)
 (261, 266,  6,  1.78555971e+00, 0.99750255,   17, 1.700e+01)
 (262, 263, 14,  1.10540837e-02, 0.68403844,   11, 1.100e+01)
 ( -1,  -1, -2, -2.00000000e+00, 0.        ,    8, 8.000e+00)
 (264, 265,  5,  6.99498212e-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.        ,    6, 6.000e+00)
 (268, 273,  4,  1.48468268e+00, 0.24229219,   75, 7.500e+01)
 (269, 272, 14, -8.07439417e-01, 0.81127812,   12, 1.200e+01)
 (270, 271, 10,  3.45403552e-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)
 (275, 278,  1,  1.39084876e+00, 0.52255937,   51, 5.100e+01)
 (276, 277,  6,  2.53980637e+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)
 (279, 282,  0, -3.96899208e-02, 0.954434  ,    8, 8.000e+00)
 (280, 281, 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[30]: 283

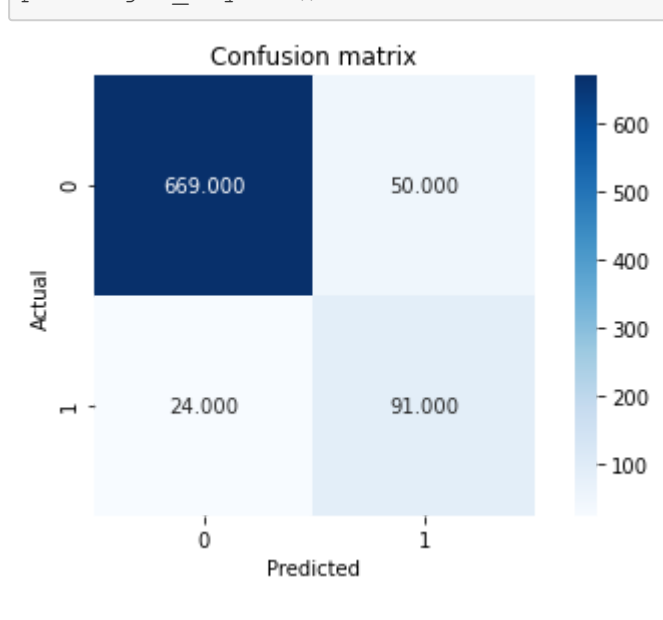
Question 01 (of 06): How many nodes are in the tree? Which variable was selected to split the root node of the tree? What can you conclude from observing the structure of the tree?

Answer: The total of 283 nodes. 'Total day minutes' was selected to split the root node of the tree. The decision tree has large number of nodes and could potential cause the over fitting

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

In [31]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [32]:
```
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 [33]:
```
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([0.96536797, 0.64539007]), array([0.93045897, 0.79130
435]), array([0.94759207, 0.7109375 ])]
```

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

Answer: The model is 91% accuracy observation were classified correctly. Based on F1 score, the classifier measure class 0 better than class 1

Part 2: k-Nearest Neighbors

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

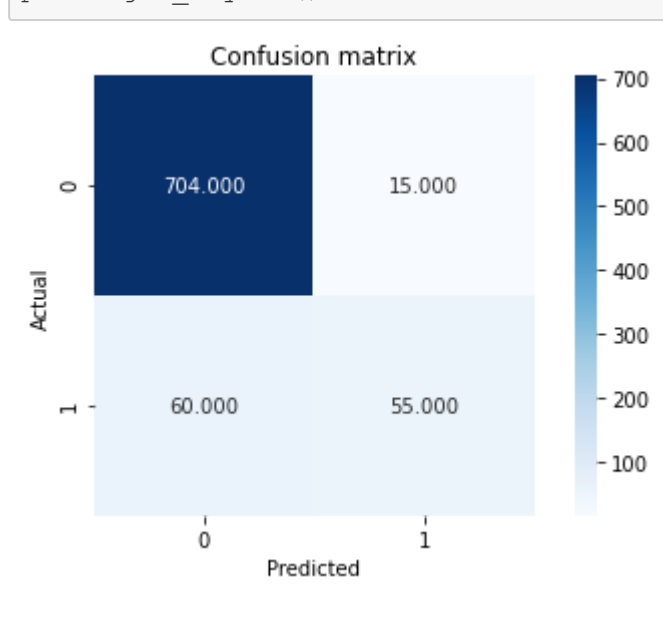
In [34]:
```
classifier = KNeighborsClassifier(n_neighbors = 3)
classifier.fit(x_train_scaled, y_train)
```

Out[34]:
```
KNeighborsClassifier(n_neighbors=3)
```

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 [35]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [36]:
```
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 [37]:
```
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]), array([0.97913769, 0.47826
087]), array([0.94942684, 0.59459459])]
```

Question 03 (of 06): What can you conclude about the performance of the k-nearest neighbors classifier? How does it compare to the previous classifier?

Answer: The model is 91% accuracy observation were classified correctly. The precison and recall of class 1 are low, and based on F1 score the classifier measure class 0 better than class 1

Part 3: Naive Bayes

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

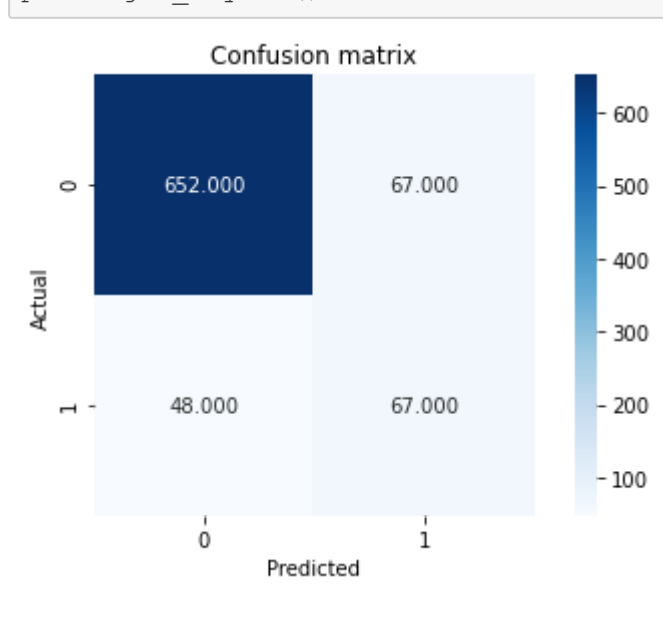
In [38]:
```
classifier = GaussianNB()
classifier.fit(x_train_scaled, y_train)
```

Out[38]:
```
GaussianNB()
```

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

In [39]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [40]:
```
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 [41]:
```
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       ]), array([0.90681502, 0.582608
7 ]), array([0.91895701, 0.53815261])]
```

Question 04 (of 06): What can you conclude about the performance of the Naive Bayes classifier? How does it compare to the previous classifiers?

Answer: The model is 86% accuracy observation were classified correctly. The precison and recall of class 1 are very low, and based on F1 score the classifier measure class 0 better than class 1

Part 4: Support Vector Machines

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

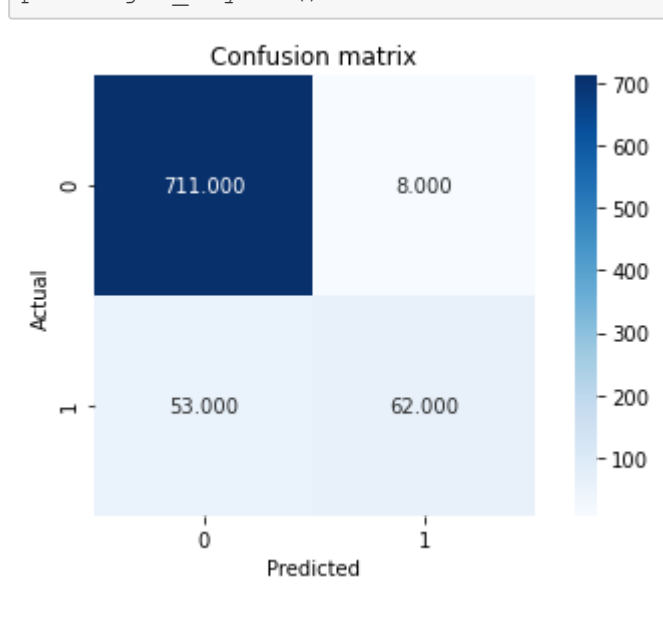
In [42]:
```
classifier = SVC(kernel = 'rbf')
classifier.fit(x_train_scaled, y_train)
```

Out[42]:
```
SVC()
```

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

In [43]:
```
y_pred = classifier.predict(x_test_scaled)
```

In [44]:
```
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 [45]:
```
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]), array([0.98887344, 0.53913
043]), array([0.95886716, 0.67027027])]
```

Question 05 (of 06): What can you conclude about the performance of the SVM classifier? How does it compare to the previous classifiers?

Answer: The model is 92% accuracy observation were classified correctly. The recall of class 1 is very low, and based on F1 score the classifier measure class 0 better than class 1

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

Answer: Based on F1 score, the decision tree classifier had the best perforance