Лабораторна робота №0.
Використання бібліотечних функцій дерева рішень

Повторіть лекційний матеріал слідуючи інструкціям

In [1]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

Спочатку налаштуємо доступ до даних на google drive (якщо ви відкриваєте блокнот в google colab, а не на PC) шляхом монтування google drive

In [2]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Перевіримо шлях до папки з матеріалами лабораторної роботи на google drive. Якщо у вас шлях відрізняється то відредагуйте

In [3]:
```
!ls gdrive/'My Drive'/TEACHING/IntroDataScience/intro_to_data_science/Lab_5_6/da
```

```
adult.data.csv   adult_test.csv   adult_train.csv   telecom_churn.csv
```

Перемістимо матеріали лабораторної роботи з google drive на віртуальну машину google colab

In [45]:
```
!cp -a gdrive/'My Drive'/TEACHING/IntroDataScience/intro_to_data_science/Lec_5_6
!ls
```

```
data     img                    lec6_linear_models.ipynb   telecom_tree2.dot
gdrive   lec5_trees_knn.ipynb   sample_data                telecom_tree2.png
```

In [46]:
```
data = pd.read_csv('data/telecom_churn.csv')
```

In [47]:
```
data.head()
```

Out[47]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122

In [48]:
```
data.drop(['State', 'Voice mail plan'], axis=1, inplace=True)
```

In [49]:
```
data['International plan'] = data['International plan'].map({'Yes': 1, 'No': 0})
```

In [50]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Account length          3333 non-null   int64  
 1   Area code               3333 non-null   int64  
 2   International plan      3333 non-null   int64  
 3   Number vmail messages   3333 non-null   int64  
 4   Total day minutes       3333 non-null   float64
 5   Total day calls         3333 non-null   int64  
 6   Total day charge        3333 non-null   float64
 7   Total eve minutes       3333 non-null   float64
 8   Total eve calls         3333 non-null   int64  
 9   Total eve charge        3333 non-null   float64
 10  Total night minutes     3333 non-null   float64
 11  Total night calls       3333 non-null   int64  
 12  Total night charge      3333 non-null   float64
 13  Total intl minutes      3333 non-null   float64
 14  Total intl calls        3333 non-null   int64  
 15  Total intl charge       3333 non-null   float64
 16  Customer service calls  3333 non-null   int64  
 17  Churn                   3333 non-null   bool   
dtypes: bool(1), float64(8), int64(9)
memory usage: 446.0 KB
```

In [51]:
```
y = data['Churn'].astype('int')
```

In [13]:
```
X = data.drop('Churn', axis=1)
```

In [52]:
```
X.shape, y.shape
```

Out[52]:
```
((3333, 17), (3333,))
```

In [53]:
```
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
```

In [54]:
```
X_train, X_valid, y_train, y_valid = train_test_split(X, y,
                                                      test_size=0.3,
                                                      random_state=17)
```

In [55]:
```
X_train.shape, X_valid.shape
```

Out[55]:
```
((2333, 17), (1000, 17))
```

In [56]:
```
first_tree = DecisionTreeClassifier(random_state=17)
```

In [57]:
```
np.mean(cross_val_score(first_tree, X_train, y_train, cv=5))
```

Out[57]:
```
0.9138423504976518
```

In [58]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [59]:
```
first_knn = KNeighborsClassifier()
```

In [60]:
```
np.mean(cross_val_score(first_knn, X_train, y_train, cv=5))
```

Out[60]:
```
0.8671274043984523
```

Налаштовуємо max_depth для дерева

In [61]:
```
from sklearn.model_selection import GridSearchCV
```

In [62]:
```
tree_params = {'max_depth': np.arange(1, 11), 'max_features':[.5, .7, 1]}
```

In [63]:
```
tree_grid = GridSearchCV(first_tree, tree_params, cv=5, n_jobs=-1)
```

In [64]:

```
%%time
tree_grid.fit(X_train, y_train);
```

```
CPU times: user 340 ms, sys: 40 ms, total: 380 ms
Wall time: 2.51 s
```

Out[64]:

```
GridSearchCV(cv=5, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              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='deprecated',
                                              random_state=17,
                                              splitter='best'),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': array([ 1,  2,  3,  4,  5,  6,  7,  8,
9, 10]),
                         'max_features': [0.5, 0.7, 1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [65]:

```
tree_grid.best_score_, tree_grid.best_params_
```

Out[65]:

```
(0.9391366681677404, {'max_depth': 6, 'max_features': 0.7})
```

In [66]:

```
knn_params = {'n_neighbors': range(5, 30, 5) }#+ list(range(50, 100, 10))}
```

In [67]:

```
knn_grid = GridSearchCV(first_knn, knn_params, cv=5)
```

In [68]:

```
%%time
knn_grid.fit(X_train, y_train);
```

```
CPU times: user 945 ms, sys: 0 ns, total: 945 ms
Wall time: 957 ms
```

Out[68]:

```
GridSearchCV(cv=5, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=None,
             param_grid={'n_neighbors': range(5, 30, 5)},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [69]:
```
knn_grid.best_score_, knn_grid.best_params_
```

Out[69]:
```
(0.8701289391697531, {'n_neighbors': 10})
```

In [70]:
```
tree_valid_pred = tree_grid.predict(X_valid)
```

In [71]:
```
from sklearn.metrics import accuracy_score
```

In [72]:
```
accuracy_score(y_valid, tree_valid_pred)
```

Out[72]:
```
0.936
```

In [73]:
```
from sklearn.metrics import confusion_matrix
```

In [74]:
```
confusion_matrix(y_valid, tree_valid_pred)
```

Out[74]:
```
array([[858,   9],
       [ 55,  78]])
```

In [75]:
```
np.bincount(y_valid)
```

Out[75]:
```
array([867, 133])
```

In [76]:
```
from sklearn.tree import export_graphviz
```

In [77]:
```
second_tree = DecisionTreeClassifier(max_depth=3).fit(X_train, y_train)
second_tree.score(X_valid, y_valid)
```

Out[77]:
```
0.905
```

In [78]:
```
export_graphviz(second_tree, out_file='telecom_tree2.dot', feature_names=X.colum
```

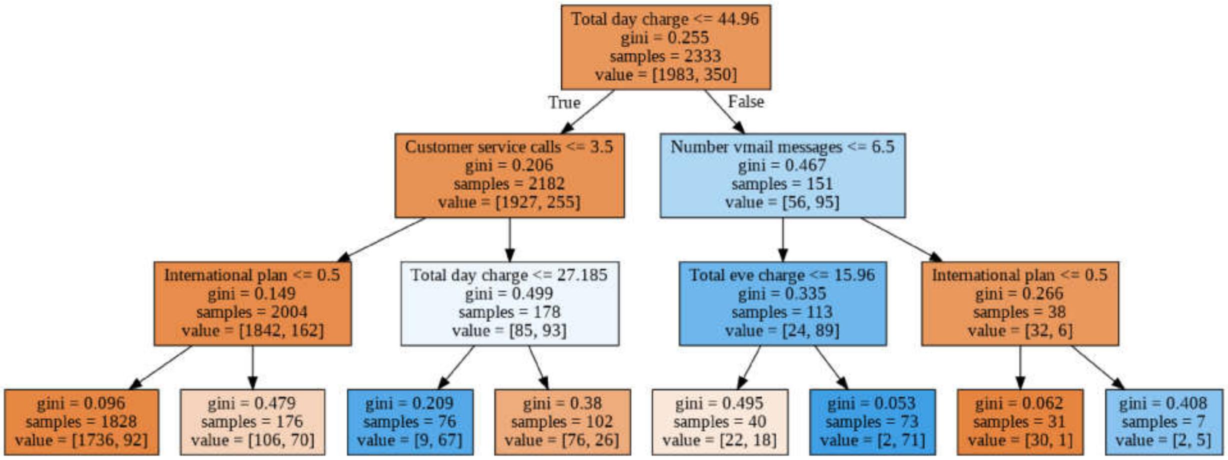
In [79]:
```
!ls -l *.png
```

```
-rw-r--r-- 1 root root 124529 Sep 19 21:06 telecom_tree2.png
```

In [81]:
```
!dot -Tpng telecom_tree2.dot -o img/telecom_tree2.png

from IPython.display import Image
Image('img/telecom_tree2.png', width=640, height=480)
```

Out[81]:


Total day charge <= 44.96
gini = 0.255
samples = 2333
value = [1983, 350]
True
False
Customer service calls <= 3.5
gini = 0.206
samples = 2182
value = [1927, 255]
Number vmail messages <= 6.5
gini = 0.467
samples = 151
value = [56, 95]
International plan <= 0.5
gini = 0.149
samples = 2004
value = [1842, 162]
Total day charge <= 27.185
gini = 0.499
samples = 178
value = [85, 93]
Total eve charge <= 15.96
gini = 0.335
samples = 113
value = [24, 89]
International plan <= 0.5
gini = 0.266
samples = 38
value = [32, 6]
gini = 0.096
samples = 1828
value = [1736, 92]
gini = 0.479
samples = 176
value = [106, 70]
gini = 0.209
samples = 76
value = [9, 67]
gini = 0.38
samples = 102
value = [76, 26]
gini = 0.495
samples = 40
value = [22, 18]
gini = 0.053
samples = 73
value = [2, 71]
gini = 0.062
samples = 31
value = [30, 1]
gini = 0.408
samples = 7
value = [2, 5]