In [3]:
```
conda install nbconvert
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/ubuntu/anaconda3

  added / updated specs:
    - nbconvert


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.11.0               |   py39h06a4308_0        14.4 MB
    ------------------------------------------------------------
                                           Total:        14.4 MB

The following packages will be UPDATED:

  conda                               4.10.3-py39h06a4308_0 --> 4.11.0-py39h06a4308_0



Downloading and Extracting Packages
conda-4.11.0         | 14.4 MB   | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

# Click through Prediction with Decision Tree classifier

In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
%config InlineBackend.figure_format = 'svg'

import warnings
import gc
import time
import itertools
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, log_loss, roc_auc_score
```

## Get dataset from kaggle

In [269…
```
! mkdir ~/.kaggle
```

```
mkdir: cannot create directory ‘/home/ubuntu/.kaggle’: File exists
```

In [3]:
```
! cp kaggle.json ~/.kaggle/
```

In [4]:
```
! chmod 600 ~/.kaggle/kaggle.json
```

In [6]:
```
!pip install --upgrade --force-reinstall --no-deps kaggle
```

```
Collecting kaggle
  Downloading kaggle-1.5.12.tar.gz (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.0/59.0 KB 3.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: kaggle
  Building wheel for kaggle (setup.py) ... done
  Created wheel for kaggle: filename=kaggle-1.5.12-py3-none-any.whl size=73051 sha256=11dd94afc337c1ddbb943d0b785bf2
```

```
3f7047046accb50956f096af21ec434361
  Stored in directory: /home/ubuntu/.cache/pip/wheels/ac/b2/c3/fa4706d469b5879105991d1c8be9a3c2ef329ba9fe2ce5085e
Successfully built kaggle
Installing collected packages: kaggle
Successfully installed kaggle-1.5.12
```

In [270…]

```
!kaggle competitions download -c avazu-ctr-prediction
```

```
Downloading avazu-ctr-prediction.zip to /home/ubuntu/jupyter/Chaeeun/Others
100%|████████████████████████████████████| 1.19G/1.19G [01:07<00:00, 24.6MB/s]
100%|████████████████████████████████████| 1.19G/1.19G [01:07<00:00, 19.0MB/s]
```

load test data

In [271…]

```
!unzip avazu-ctr-prediction.zip
```

```
Archive:  avazu-ctr-prediction.zip
  inflating: sampleSubmission.gz
  inflating: test.gz
  inflating: train.gz
```

In [272…]

```
test_file = "test.gz"
test = pd.read_csv(test_file, compression='gzip')
```

In [359…]

```
from PIL import Image

Image.open("feature_description.JPG")
```

Out[359…]

- id: ad identifier, such as 10000094181510942 73, 10000169349117863715
- click: 0 for non-click, 1 for click
- hour: in the format of YYMMDDHH, for example, 14102100
- C1: anonymized categorical variable, such as 1005, 1002
- banner_pos: where a banner is located, 1 and 0
- site_id: site identifier, such as 1fbe01fe, fe8cc448, d6137915
- site_domain: hashed site domain, such as 'bb1ef334', 'f3845767
- site_category: hashed site category, such as 28905ebd, 28905ebd
- app_id: mobile app identifier
- app_domain
- app_category
- device_id: mobile device identifier
- device_ip: IP address
- device_model: such as iPhone 6, Samsung, hashed by the way
- device_type: such as tablet, smartphone, hashed
- device_conn_type: Wi-Fi or 3G for example, again hashed in the data
- C14-C21: anonymized categorical variables

In [273…]

```
test.head()
```

Out[273…]

| | id | hour | C1 | banner_pos | site_id | site_domain | site_category | app_id | app_domain | app_category | ... | de |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.000017e+19 | 14103100 | 1005 | 0 | 235ba823 | f6ebf28e | f028772b | ecad2386 | 7801e8d9 | 07d7df22 | ... | |
| **1** | 1.000018e+19 | 14103100 | 1005 | 0 | 1fbe01fe | f3845767 | 28905ebd | ecad2386 | 7801e8d9 | 07d7df22 | ... | |
| **2** | 1.000055e+19 | 14103100 | 1005 | 0 | 1fbe01fe | f3845767 | 28905ebd | ecad2386 | 7801e8d9 | 07d7df22 | ... | |
| **3** | 1.000109e+19 | 14103100 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 51cedd4e | aefc06bd | 0f2161f8 | ... | |
| **4** | 1.000138e+19 | 14103100 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 9c13b419 | 2347f47a | f95efa07 | ... | |

5 rows × 23 columns

size of train dataset is very big, so need to divide into chunks

In [274…]

```
train_file = "train.gz"
reader = pd.read_csv(train_file, chunksize=10**6, iterator=True)

train = pd.DataFrame()
start = time.time()

for i, chunk in enumerate(reader):
    chunk = chunk.sample(frac=.65, replace=False, random_state=516)
```

```
    neg_samp = chunk[chunk['click'] == 0].sample(n=len(chunk[chunk['click'] == 1]), random_state=2021)
    train = pd.concat([train, neg_samp, chunk[chunk['click'] == 1]], axis=0)
    if i % 20 == 0:
        print('Processing Chunk No. {}'.format(i))

print('the program costs %.2f seconds'%(time.time() - start))

del neg_samp
gc.collect()

print('train has {} rows and {} columns'.format(train.shape[0], train.shape[1]))
print('test has {} rows and {} columns'.format(test.shape[0], test.shape[1]))
```

```
Processing Chunk No. 0
Processing Chunk No. 20
Processing Chunk No. 40
the program costs 213.54 seconds
train has 8925858 rows and 24 columns
test has 4577464 rows and 23 columns
```

In [275…

```
train.head()
```

Out[275…

| | id | click | hour | C1 | banner_pos | site_id | site_domain | site_category | app_id | app_domain | ... | devi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **740545** | 1.521976e+19 | 0 | 14102104 | 1005 | 1 | e023ba3e | 75f9ddc3 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **491413** | 1.237389e+19 | 0 | 14102103 | 1005 | 1 | d9750ee7 | 98572c79 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **771247** | 1.717520e+19 | 0 | 14102104 | 1005 | 1 | b7e9786d | b12b9f85 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **348689** | 1.739074e+19 | 0 | 14102102 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 795a164e | 2347f47a | ... | |
| **158412** | 1.475080e+19 | 0 | 14102101 | 1005 | 0 | 543a539e | c7ca3108 | 3e814130 | ecad2386 | 7801e8d9 | ... | |

5 rows × 24 columns

In [276…

```
train.reset_index(drop=True)
```

Out[276…

| | id | click | hour | C1 | banner_pos | site_id | site_domain | site_category | app_id | app_domain | ... | dev |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.521976e+19 | 0 | 14102104 | 1005 | 1 | e023ba3e | 75f9ddc3 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **1** | 1.237389e+19 | 0 | 14102103 | 1005 | 1 | d9750ee7 | 98572c79 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **2** | 1.717520e+19 | 0 | 14102104 | 1005 | 1 | b7e9786d | b12b9f85 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **3** | 1.739074e+19 | 0 | 14102102 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 795a164e | 2347f47a | ... | |
| **4** | 1.475080e+19 | 0 | 14102101 | 1005 | 0 | 543a539e | c7ca3108 | 3e814130 | ecad2386 | 7801e8d9 | ... | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **8925853** | 1.312272e+19 | 1 | 14103023 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 9c13b419 | 2347f47a | ... | |
| **8925854** | 5.040032e+18 | 1 | 14103023 | 1005 | 1 | e151e245 | 7e091613 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **8925855** | 8.002520e+18 | 1 | 14103021 | 1005 | 0 | 4bf5bbe2 | 6b560cc1 | 28905ebd | ecad2386 | 7801e8d9 | ... | |
| **8925856** | 2.634610e+18 | 1 | 14103021 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 9c13b419 | 2347f47a | ... | |
| **8925857** | 8.702383e+18 | 1 | 14103020 | 1005 | 0 | 6c7e709c | 246937e7 | 3e814130 | ecad2386 | 7801e8d9 | ... | |

8925858 rows × 24 columns

In [277…

```
train.head()
```

Out[277…

| | id | click | hour | C1 | banner_pos | site_id | site_domain | site_category | app_id | app_domain | ... | devi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **740545** | 1.521976e+19 | 0 | 14102104 | 1005 | 1 | e023ba3e | 75f9ddc3 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **491413** | 1.237389e+19 | 0 | 14102103 | 1005 | 1 | d9750ee7 | 98572c79 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **771247** | 1.717520e+19 | 0 | 14102104 | 1005 | 1 | b7e9786d | b12b9f85 | f028772b | ecad2386 | 7801e8d9 | ... | |
| **348689** | 1.739074e+19 | 0 | 14102102 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 795a164e | 2347f47a | ... | |
| **158412** | 1.475080e+19 | 0 | 14102101 | 1005 | 0 | 543a539e | c7ca3108 | 3e814130 | ecad2386 | 7801e8d9 | ... | |

5 rows × 24 columns

## shuffle data

In [278…

```
from sklearn.utils import shuffle
train = shuffle(train)
```

```
test = shuffle(test)
```

In [279…
```
train.head()
```

Out[279…

| | id | click | hour | C1 | banner_pos | site_id | site_domain | site_category | app_id | app_domain | … | de |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **28739433** | 1.775597e+19 | 1 | 14102809 | 1005 | 1 | d9750ee7 | 98572c79 | f028772b | ecad2386 | 7801e8d9 | … | |
| **7000348** | 4.609652e+17 | 1 | 14102211 | 1005 | 0 | 1fbe01fe | f3845767 | 28905ebd | ecad2386 | 7801e8d9 | … | |
| **37706442** | 9.145271e+18 | 1 | 14103008 | 1005 | 0 | 325dbe14 | 373cce89 | f028772b | ecad2386 | 7801e8d9 | … | |
| **17948515** | 4.297616e+18 | 1 | 14102511 | 1005 | 1 | 5b4d2eda | 16a36ef3 | f028772b | ecad2386 | 7801e8d9 | … | |
| **18755767** | 1.372269e+19 | 1 | 14102515 | 1005 | 0 | 6256f5b4 | 28f93029 | f028772b | ecad2386 | 7801e8d9 | … | |

5 rows × 24 columns

In [280…
```
train.columns
```

Out[280…
```
Index(['id', 'click', 'hour', 'C1', 'banner_pos', 'site_id', 'site_domain',
       'site_category', 'app_id', 'app_domain', 'app_category', 'device_id',
       'device_ip', 'device_model', 'device_type', 'device_conn_type', 'C14',
       'C15', 'C16', 'C17', 'C18', 'C19', 'C20', 'C21'],
      dtype='object')
```

In [281…
```
len(train.columns)
```

Out[281… 24

In [282…
```
test.head()
```

Out[282…

| | id | hour | C1 | banner_pos | site_id | site_domain | site_category | app_id | app_domain | app_category |
|---|---|---|---|---|---|---|---|---|---|---|
| **1154107** | 5.292171e+18 | 14103107 | 1005 | 1 | 5b4d2eda | 16a36ef3 | f028772b | ecad2386 | 7801e8d9 | 07d7df22 |
| **3923939** | 3.486957e+18 | 14103118 | 1005 | 0 | 17d1b03f | f3845767 | f028772b | ecad2386 | 7801e8d9 | 07d7df22 |
| **3778710** | 9.250585e+17 | 14103117 | 1005 | 0 | 6256f5b4 | 28f93029 | f028772b | ecad2386 | 7801e8d9 | 07d7df22 |
| **1234352** | 9.869577e+18 | 14103107 | 1005 | 0 | b99a2c43 | cc962a1f | f028772b | ecad2386 | 7801e8d9 | 07d7df22 |
| **3433745** | 1.767858e+19 | 14103116 | 1005 | 0 | 85f751fd | c4e18dd6 | 50e219e0 | 92f5800b | ae637522 | 0f2161f8 |

5 rows × 23 columns

In [283…
```
train.to_csv("train.csv", index = False)
test.to_csv("test.csv", index = False)
```

# Read data in a dictionary

In [303…
```
import csv

def read_ad_click_data(n, offset=0):
    X_dict, y = [], []
    with open('train.csv', 'r') as csvfile:
        reader = csv.DictReader(csvfile)
        for i in range(offset):
            next(reader)
        i = 0
        for row in reader:
            i += 1
            y.append(int(row['click']))
            del row['click'], row['id'], row['hour'], row['device_id'], row['device_ip']#remove irrelevant features
            X_dict.append(dict(row))
            if i >= n:
                break
    return X_dict, y
```

In [304…
```
n = 100000#only use 100000 training samples
X_dict_train, y_train = read_ad_click_data(n)
```

In [305…
```
print(X_dict_train[0])#first training sample (hashed values for privacy)
print(X_dict_train[1])#second training sample (hashed values for privacy)
```

```
{'C1': '1005', 'banner_pos': '1', 'site_id': 'd9750ee7', 'site_domain': '98572c79', 'site_category': 'f028772b', 'ap
p_id': 'ecad2386', 'app_domain': '7801e8d9', 'app_category': '07d7df22', 'device_model': 'd6e0e6ff', 'device_type':
'1', 'device_conn_type': '0', 'C14': '23144', 'C15': '320', 'C16': '50', 'C17': '2665', 'C18': '0', 'C19': '34', 'C2
0': '100070', 'C21': '221'}
{'C1': '1005', 'banner_pos': '0', 'site_id': '1fbe01fe', 'site_domain': 'f3845767', 'site_category': '28905ebd', 'ap
p_id': 'ecad2386', 'app_domain': '7801e8d9', 'app_category': '07d7df22', 'device_model': '1f0bc64f', 'device_type':
'1', 'device_conn_type': '0', 'C14': '15708', 'C15': '320', 'C16': '50', 'C17': '1722', 'C18': '0', 'C19': '35', 'C2
0': '-1', 'C21': '79'}
```

In [306…
```
len(X_dict_train)#length of dictionary
```

Out[306…
```
100000
```

In [307…
```
len(X_dict_train[0])#number of attributes(==textbook)
```

Out[307…
```
19
```

In [308…
```
len(y_train)
```

Out[308…
```
100000
```

# Encode Data

**DictVectorizer: converts a categorical feature with k possible values into k binary features.**

ex.)

In [52]:
```
from sklearn.feature_extraction import DictVectorizer
#example usage of DictVectorizer for one-hot encoding
sample = [{'ch':'-1', 'a':'0.2','ee':'hello'},{'ch':'0','a':'2'}, {'ee': 'hi'}]

tmp_encoder = DictVectorizer(sparse = False)
sample_d = tmp_encoder.fit_transform(sample)
```

In [53]:
```
sample_d# ch==-1 | ch==0 | a == 0.2 | a == 2 | ee==hello | ee == hi
```

Out[53]:
```
array([[1., 0., 1., 0., 1., 0.],
       [0., 1., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 1.]])
```

In [309…
```
from sklearn.feature_extraction import DictVectorizer
dict_one_hot_encoder = DictVectorizer(sparse=False)
X_train = dict_one_hot_encoder.fit_transform(X_dict_train)
```

In [310…
```
len(X_train)
```

Out[310…
```
100000
```

In [311…
```
len(X_train[0])
```

Out[311…
```
9694
```

In [312…
```
X_train[0]#first training datasample
```

Out[312…
```
array([0., 0., 0., ..., 0., 0., 0.])
```

In [313…
```
X_dict_test, y_test = read_ad_click_data(n, n)
```

In [314…
```
X_test = dict_one_hot_encoder.transform(X_dict_test)
```

In [315…
```
X_test[0]#first testing datasample
```

Out[315…
```
array([0., 0., 0., ..., 0., 0., 0.])
```

In [316…

```
len(X_test)
```

Out[316… 100000

In [317…
```
print(len(X_test[0]))
```

9694

Difference exists for values of len(X_train[0]), len(X_test[0]) between this code output, and the textbook. It is because of the code for reading training sample is different( -> training dataset in the textbook != training dataset in this code)

## Train a Decision Tree Classifier

In [326…
```
from sklearn.tree import DecisionTreeClassifier
parameters = {'max_depth': [2, 3, 10, None]}#try training with max_depth with different values
decision_tree = DecisionTreeClassifier(criterion='gini', min_samples_split=30)
```

In [327…
```
from sklearn.model_selection import GridSearchCV#Exhaustive search over specified parameter values for an estimator.
grid_search = GridSearchCV(decision_tree, parameters, cv=3, scoring='roc_auc')
```

In [328…
```
grid_search.fit(X_train, y_train)#train
```

Out[328…
```
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(min_samples_split=30),
             param_grid={'max_depth': [2, 3, 10, None]}, scoring='roc_auc')
```

In [329…
```
grid_search.best_estimator_
```

Out[329… `DecisionTreeClassifier(max_depth=10, min_samples_split=30)`

find best parameter

In [330…
```
print(grid_search.best_params_)
```

```
{'max_depth': 10}
```

In [331…
```
from pylab import *
decision_tree_best = grid_search.best_estimator_#get the best classifier
pos_prob = decision_tree_best.predict_proba(X_test)[:, 1]#predict
```

in the text book: 0.692

In [336…
```
from sklearn.metrics import roc_auc_score
print('The ROC AUC on testing set is: {0:.3f}'.format(roc_auc_score(y_test, pos_prob)))
```

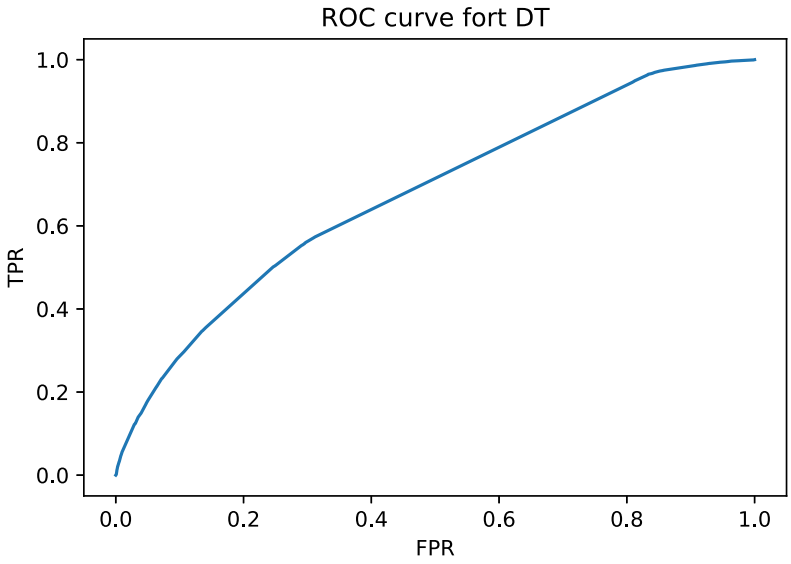
```
The ROC AUC on testing set is: 0.676
```

In [185…
```
pos_prob
```

Out[185…
```
array([0.41407185, 0.17550159, 0.13790859, ..., 0.87188453, 0.61865615,
       0.76803442])
```

In [364…
```
from sklearn.metrics import roc_curve
from matplotlib import pyplot as plt
fpr, tpr, _ = roc_curve(y_test, pos_prob)

plt.clf()
plt.plot(fpr, tpr,label="DT")
plt.plot()
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve fort DT')
plt.show()
```




# Random Forest

Hyper parameters of Random Forest:

- max_features: The number of features to consider at each best splitting point search.
- n_estimators: The number of trees considered for majority voting
- min_samples_split: Minimal number of samples required for further split at a node.(ex. 10,30,50,...)

In [346…
```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100, criterion='gini', min_samples_split=30, n_jobs=-1)
grid_search = GridSearchCV(random_forest, parameters, cv=3, scoring='roc_auc')
```

In [347…
```
grid_search.fit(X_train, y_train)
```

Out[347…
```
GridSearchCV(cv=3,
             estimator=RandomForestClassifier(min_samples_split=30, n_jobs=-1),
             param_grid={'max_depth': [2, 3, 10, None]}, scoring='roc_auc')
```

In [348…
```
print(grid_search.best_params_)
print(grid_search.best_score_)
```

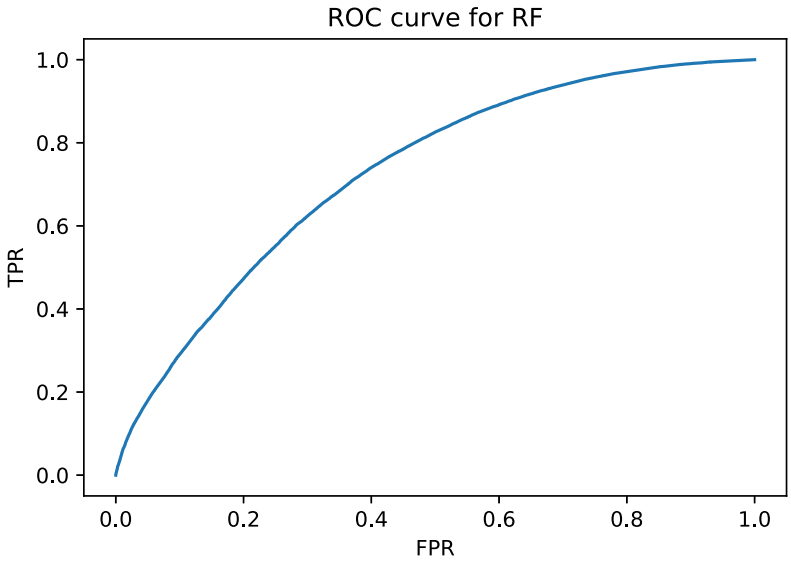
```
{'max_depth': None}
0.7258407193811719
```

In [349…
```
from sklearn.metrics import roc_auc_score
random_forest_best = grid_search.best_estimator_
pos_prob_rf = random_forest_best.predict_proba(X_test)[:, 1]
print(f'The ROC AUC on testing set is: {roc_auc_score(y_test, pos_prob_rf):.3f}')
```

```
The ROC AUC on testing set is: 0.729
```

In [363…
```
from sklearn.metrics import roc_curve
from matplotlib import pyplot as plt
fpr, tpr, _ = roc_curve(y_test, pos_prob_rf)

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve for RF')
plt.show()
```




# Boost

- XGBoost
- AdaBoost
- LightGBM

In [188…

```
pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-1.5.2-py3-none-manylinux2014_x86_64.whl (173.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 173.6/173.6 MB 9.2 MB/s eta 0:00:00m eta 0:00:01[36m0:00:01
Requirement already satisfied: scipy in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from xgboost) (1.7.1)
Requirement already satisfied: numpy in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from xgboost) (1.20.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.5.2
Note: you may need to restart the kernel to use updated packages.
```

In [352…

```
import xgboost as xgb
model = xgb.XGBClassifier(learning_rate=0.1, max_depth=10, n_estimators=1000)
```

In [353…

```
model.fit(X_train, y_train, verbose = 0)
```

```
[12:11:30] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the o
bjective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restor
e the old behavior.
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipykernel_18626/1345923981.py in <module>
----> 1 model.fit(X_train, y_train, verbose = 0)

~/anaconda3/lib/python3.9/site-packages/xgboost/core.py in inner_f(*args, **kwargs)
    504         for k, arg in zip(sig.parameters, args):
    505             kwargs[k] = arg
--> 506         return f(**kwargs)
    507
    508     return inner_f

~/anaconda3/lib/python3.9/site-packages/xgboost/sklearn.py in fit(self, X, y, sample_weight, base_margin, eval_set,
 eval_metric, early_stopping_rounds, verbose, xgb_model, sample_weight_eval_set, base_margin_eval_set, feature_weigh
ts, callbacks)
   1248         )
   1249
-> 1250         self._Booster = train(
   1251             params,
   1252             train_dmatrix,

~/anaconda3/lib/python3.9/site-packages/xgboost/training.py in train(params, dtrain, num_boost_round, evals, obj, fe
val, maximize, early_stopping_rounds, evals_result, verbose_eval, xgb_model, callbacks)
    186     Booster : a trained booster model
    187     """
--> 188     bst = _train_internal(params, dtrain,
    189                           num_boost_round=num_boost_round,
    190                           evals=evals,

~/anaconda3/lib/python3.9/site-packages/xgboost/training.py in _train_internal(params, dtrain, num_boost_round, eval
s, obj, feval, xgb_model, callbacks, evals_result, maximize, verbose_eval, early_stopping_rounds)
```

```
     79         if callbacks.before_iteration(bst, i, dtrain, evals):
     80             break
---> 81         bst.update(dtrain, i, obj)
     82         if callbacks.after_iteration(bst, i, dtrain, evals):
     83             break

~/anaconda3/lib/python3.9/site-packages/xgboost/core.py in update(self, dtrain, iteration, fobj)
   1678
   1679         if fobj is None:
-> 1680             _check_call(_LIB.XGBoosterUpdateOneIter(self.handle,
   1681                                                     ctypes.c_int(iteration),
   1682                                                     dtrain.handle))

KeyboardInterrupt:
```

In [ ]:
```
pos_prob_xgb = model.predict_proba(X_test)[:, 1]
from sklearn.metrics import roc_auc_score
print(f'The ROC AUC on testing set is: {roc_auc_score(y_test, pos_prob_xgb):.3f}')
```

## AdaBoost

### Hyper parameters of AdaBoost:

- learning rate: Weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier.
- n_estimators: The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.

In [ ]:
```
from sklearn.ensemble import AdaBoostClassifier
abc = AdaBoostClassifier(learning_rate=0.1, n_estimators=100)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)
```

In [ ]:
```
pos_prob_ada = model.predict_proba(X_test)[:, 1]
from sklearn.metrics import roc_auc_score
print(f'The ROC AUC on testing set is: {roc_auc_score(y_test, pos_prob_ada):.3f}')
```

## LightGBM

In [341…
```
pip install lightgbm
```

```
Collecting lightgbm
  Downloading lightgbm-3.3.2-py3-none-manylinux1_x86_64.whl (2.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 23.0 MB/s eta 0:00:0031m28.5 MB/s eta 0:00:01
Requirement already satisfied: scikit-learn!=0.22.0 in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from ligh
tgbm) (0.24.2)
Requirement already satisfied: numpy in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from lightgbm) (1.20.3)
Requirement already satisfied: wheel in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from lightgbm) (0.37.0)
Requirement already satisfied: scipy in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from lightgbm) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from scik
it-learn!=0.22.0->lightgbm) (2.2.0)
Requirement already satisfied: joblib>=0.11 in /home/ubuntu/anaconda3/lib/python3.9/site-packages (from scikit-lear
n!=0.22.0->lightgbm) (1.1.0)
Installing collected packages: lightgbm
Successfully installed lightgbm-3.3.2
Note: you may need to restart the kernel to use updated packages.
```

In [342…
```
from lightgbm import LGBMClassifier
lgbm = LGBMClassifier(n_estimators=200)
model = lgbm.fit(X_train, y_train)
```

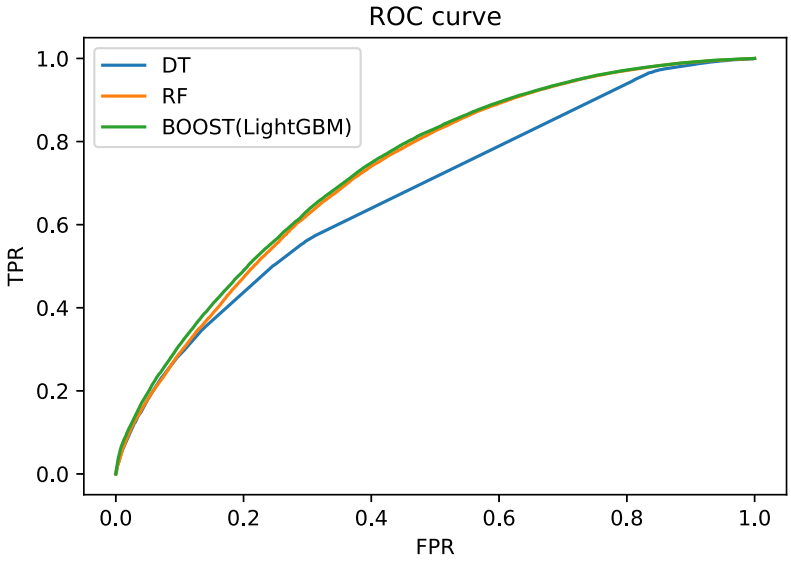
In [ ]:
```
pos_prob_lgbm = model.predict_proba(X_test)[:, 1]
from sklearn.metrics import roc_auc_score
print(f'The ROC AUC on testing set is: {roc_auc_score(y_test, pos_prob_lgbm):.3f}')
```

# Comparison of ROC for DecisionTree, Random Forest, Boosting

In [362…
```
from sklearn.metrics import roc_curve
from matplotlib import pyplot as plt

fpr, tpr, _ = roc_curve(y_test, pos_prob)
fpr1, tpr1, _ = roc_curve(y_test, pos_prob_rf)
fpr2, tpr2, _ = roc_curve(y_test, pos_prob_lgbm)
```

```
plt.clf()
plt.plot(fpr, tpr,label="DT")
plt.plot(fpr1, tpr1,label="RF")
plt.plot(fpr2, tpr2,label="BOOST(LightGBM)")
plt.legend()
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()
```




In [ ]: