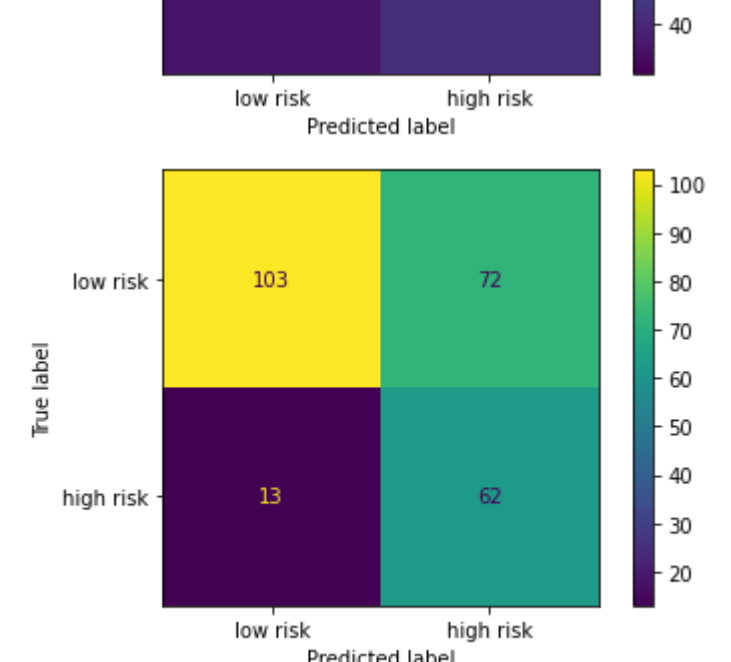
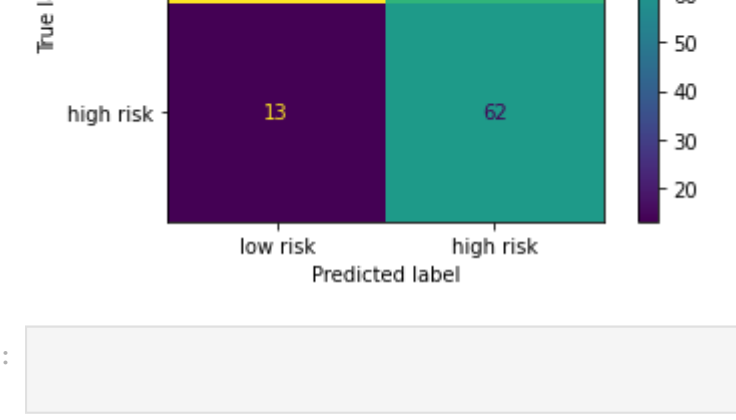
In [3]:

```python
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

#print("hello")

# need to library the package

import pandas as pd
import numpy as np
import xgboost as xgb

# need to import the necessary dataset
from sklearn.model_selection import train_test_split
from sklearn.metrics import balanced_accuracy_score, roc_auc_score, make_scorer
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.multioutput import MultiOutputRegressor


#load the data to see the dataframe looks like
german = pd.read_csv(r'german.csv')
a= german.head()
print(a)

# because the purpose question is a interview question, so we did not
# use it as a prediction in the model, we will remove it

german.drop(['Purpose'],
            axis=1, inplace=True)
b= german.head()
print(b)

# becasue numebr id is different for each individual, thus it is not that
# useful to use for classification

german.drop(['number'], axis=1, inplace= True)
c= german.head()
print(c)

# checking for missing data and identify to see what to do about it

d= german.dtypes
print(d)

# try to find the unique intepretation of the varibale and see if they make
# sense to exists

e= german['Sex'].unique()
print(e)

f= german['Housing'].unique()
print(f)

g= german['Saving accounts'].unique()
print(g)

# when you do this , it will allow you see how the catergories are, and we
# need to replace all the white spaces in all the columns with underscores.
# this is because we can draw a nicer picture of XGBoost trees later

german.replace(' ', '_', regex= True, inplace = True)
h= german.head()
print(h)
i= german['Saving accounts'].unique()
print(i)



## spliting the data to dependent variable and independent variable
# the first step is to split the data into two parts:
    #1. the column of data that we will use to make classification
    #2. the cloumn of data that we want to predict
# we will use capital X to define the columns of data that we use to make
#classifications and small y to define the thing we want to predict
# in this case we want to predict whether some one is high risk or low risk to
#return the credit that someone take in the bank according to the attribute that
#this person have in the factors

# thus, we want to predict the Default value here

X= german.drop('Default', axis=1).copy()
A= X.head()
print(A)

y= german['Default'].copy()
B= y.head()
print(B)

## Format the data part 2 : One -hot Encoding
# we are going to find the tyoe of the varible data on whether it is float or
# categorical

C= X.dtypes
print (C)

D= pd.get_dummies(X,  columns=['Sex']).head()

print(D)

X_encoded= pd.get_dummies(X,  columns=['Sex',
                                       'Housing',
                                       'Saving accounts',
                                       'Checking account'])
E= X_encoded.head()
print(E)

# we need to varify that y only contain 0 and 1 unique function

F= y.unique()
print (F)

## Build a basic Preliinary XGboost Model
# we haev successfully split the model into traiing and testing sets, we will
# build a model now.
# we discover that this data is imbalended by dividing the numebr of people
# who consider not return, where y=1, divide by the number of people in the
#dataset


j= sum(y)/len(y)
print(j)

 # we see only 30% of people consider high risk that they did not return the
 #money. thus, when we soikt the data into training and testing, we will slipt
# using stratificaion in order to maintain the same percentage of people
# who do not return the borrow money in both the traiing set and testing set.


X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
                                                    random_state= 21,
                                                    stratify=y )

k= sum(y_train)/len(y_train)
print(k)

l= sum(y_test)/len(y_test)
print(l)


# both k and l =30%, this shows that stratify works as expected that both y_test
# and y_train have the same number of high risk people profile that does not
# return money.
# we can then build the preliminary model


clf_xgb= xgb.XGBClassifier(objective = 'binary:logistic', missing = 1,
                           seed=21)
clf_xgb.fit(X_train,
            y_train,
            verbose=True,
            early_stopping_rounds=10,
            eval_metric='aucpr',
            eval_set=[(X_test, y_test)])

# we see the best iteration is 22 is 0.63548
# we will then see how the performance on the testing dataset by running
# the testing dateset down the model and drawing the confusion matrix

t1= clf_xgb.predict(X_train)
print(t1.shape)


temp = clf_xgb.predict(X_test)
print(temp.shape)


plot_confusion_matrix(clf_xgb,
                      X_test,
                      y_test,
                      values_format='d',
                      display_labels=["low risk","high risk"])



# we can see in this confusion matrix,we see that of 175 epople that
#has lower risk, 149 (85.14%) is correctly classified. and of the 75 people
#who are high risk,42 (56%) is correctly classified. so this means this xgboost
# model was not that awesome.this may becasue the imbalanced dataset in the
# begining. but becasue high risk customer profile cost a lot more money to the
# bank, we want to capture more of people who are gih risk. thus we want to
# try to improve the prediction using Cross Vlidation to optimize the parameters.


## then we want use AUC for evaluation, because grivdDearchCV takes long time
#for the optimize parameters in a short period of time.

## round 1
#
param_grid= {
    'max_depth':[3,4,5],
    'learning_rate':[0.1,0.01,0.05],
    'gamma':[0,0.25,1.0],
    'reg_lambda':[0,1.0,10.0],
    'scale_pos_weight':[1,3,5]
}

#xgbr = xgb.XGBRegressor(seed = 20)
#clf = GridSearchCV(estimator=xgbr,
#                   param_grid=param_grid,
#                   scoring='neg_mean_squared_error',
#                   verbose=1)
#clf.fit(X, y)
#print("Best parameters:", clf.best_params_)
#print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))


#gsc = GridSearchCV(
 #           estimator=xgb.XGBClassifier(),
#          param_grid={
#                'max_depth':[3,4,5],
#                'learning_rate':[0.1,0.01,0.05],
#                'gamma':[0,0.25,1.0],
#                'reg_lambda':[0,1.0,10.0],
#                'scale_pos_weight':[1,3,5],
 #               },
 #           cv=3, scoring='neg_mean_squared_error', verbose=0, n_jobs=-1)

#grid_result = MultiOutputRegressor(gsc).fit(X_train, y_train)

#best_params = grid_result.estimators_[0].best_params_

optimal_params = GridSearchCV(
    estimator=xgb.XGBClassifier(objective='binary:logistic',
                                seed=21,
                                subsample=0.9,
                                colsample_bytree=0.5),
    param_grid=param_grid,
    scoring='roc_auc',
    verbose=0,
    n_jobs=10,
    cv=3
    )

#from sklearn.metrics import fbeta_score, make_scorer
#from sklearn.metrics.roc_auc_score import roc_auc
#sorted(sklearn.metrics.SCORERS.keys()
# from sklearn.model_selection import metrics.roc_auc_score
optimal_params.fit(X_train,
                   y_train,
                   early_stopping_rounds=10,
                   eval_metric='auc',
                   eval_set=[(X_test,y_test)],
                   verbose=False)

#print(optimal_params.cv_results_)
print(optimal_params.best_params_)

#we can see the optimal output said by round one is gamma= 0.25,
# learning rate= 0.1, max__depth=3,reg_lammbda=10.0,
#scale_pos_weight=3

# if the calue is a middle vaue then we try to keep that, that is already a
#optimal in the case, then if it is a lower boudn or upper bound of that candidate
#we will test out for more values that based on that

## RROUND 2

param_grid= {
    'max_depth':[2,3,4],
    'learning_rate':[0.1, 0.5, 1],
    'gamma':[0.25],
    'reg_lambda':[10.0,20,100],
    'scale_pos_weight':[3]
}

optimal_params = GridSearchCV(
    estimator=xgb.XGBClassifier(objective='binary:logistic',
                                seed=21,
                                subsample=0.9,
                                colsample_bytree=0.5),
    param_grid=param_grid,
    scoring='roc_auc',
    verbose=0,
    n_jobs=10,
    cv=3
    )

optimal_params.fit(X_train,
                   y_train,
                   early_stopping_rounds=10,
                   eval_metric='auc',
                   eval_set=[(X_test,y_test)],
                   verbose=False)


print(optimal_params.best_params_)


# these are the round 2 resuts we have
#gamma': 0.25, 'learning_rate': 0.1, 'max_depth': 2, 'reg_lambda': 20,
# 'scale_pos_weight': 3


# now we want to build our final XGBoost model and evlaute the results

clf_xgb= xgb.XGBClassifier(seed=21,
                           objective= 'binary:logistic',
                           gamma= 0.25,
                           learn_rate=0.1,
                           max_depth=2,
                           reg_lambda=20,
                           scale_pos_weight=3,
                           subsample=0.9,
                           colsample_bytree=0.5)

clf_xgb.fit(X_train,
            y_train,
            verbose=True,
            early_stopping_rounds=10,
            eval_metric='aucpr',
            eval_set=[(X_test,y_test)])

# we have the following results:
    #XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
#                  colsample_bynode=1, colsample_bytree=0.5,
#                  enable_categorical=False, gamma=0.25, gpu_id=-1,
#                  importance_type=None, interaction_constraints='', learn_rate=0.1,
#                   learning_rate=0.300000012, max_delta_step=0, max_depth=2,
#                   min_child_weight=1, missing=nan, monotone_constraints='()',
#                   n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',
#                   random_state=21, reg_alpha=0, reg_lambda=20, scale_pos_weight=3,
#                   seed=21, subsample=0.9, tree_method='exact',
#                  validate_parameters=1, verbosity=None)



#let then plot another matrix and see how it goes,

plot_confusion_matrix(clf_xgb,
                      X_test,
                      y_test,
                      values_format='d',
                      display_labels=["low risk","high risk"])

# we can see a much better at identify poeple that is high risk.
# out of 175 people that low risk, we identify 58.9% (lower than expected),
#but for high risk people, out of 75, 62 (83%)is correctly identified,  which
#is much higher than previous date rate of 56%, evne though this come at the
#expense that not being able to correctly identify the low risk people. However,
# i think this still a better model because we did manage to do identify high
# risk poeple, because they are the potential cost more money and resourse to
# pubnish and prevent sth like that happening, whereas for the wrongly identify
# peopel we can give some for example couple and they will not be that offened
#if we identify them wrongly compared to leting go dangerous people out.


clf_xgb= xgb.XGBClassifier(seed=21,
                           objective= 'binary:logistic',
                           gamma= 0.25,
                           learn_rate=0.1,
                           max_depth=2,
                           reg_lambda=20,
                           scale_pos_weight=3,
                           subsample=0.9,
                           colsample_bytree=0.5,
                           n_estimators=1)

clf_xgb.fit(X_train, y_train)

# output: is
# XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
#              colsample_bynode=1, colsample_bytree=0.5,
#              enable_categorical=False, gamma=0.25, gpu_id=-1,
#              importance_type=None, interaction_constraints='', learn_rate=0.1,
#              learning_rate=0.300000012, max_delta_step=0, max_depth=2,
#              min_child_weight=1, missing=nan, monotone_constraints='()',
#              n_estimators=1, n_jobs=8, num_parallel_tree=1, predictor='auto',
#              random_state=21, reg_alpha=0, reg_lambda=20, scale_pos_weight=3,
#              seed=21, subsample=0.9, tree_method='exact',
#              validate_parameters=1, verbosity=None)

bst= clf_xgb.get_booster()

for importance_type in ('weight','gain','cover','total_gain','total_cover'):
  print ('%s:'% importance_type,bst.get_score(importance_type=importance_type))


node_params= {'shape':'box',
              'skyle': 'filled, rounded',
              'fillcolor': '#78cbe'}

leaf_params={'shape':'box',
             'style':'filled',
             'fillcolor':'#e4038'}

xgb.to_graphviz(clf_xgb, num_trees=0,size="10,10",
                condition_node_params=node_params,
                leaf_node_params=leaf_params)

# save as individual pdf

graph_data= xgb.to_graphviz(clf_xgb,num_tree=0,size="10,10",
                            condition_node_params=node_params,
                            leaf_node_params=leaf_params)

graph_data.view(filename='xgboost_tree_risk_customer')

#interpreting the tree

#each level give you a choice of yes and no, it helps you determine which way
#you want to go, then the leaf indicate the probability for each action you take.
```

```
   number  Age     Sex  Job Housing Saving accounts Checking account  \
0       0   67    male    2     own             NaN           little
1       1   22  female    2     own          little         moderate
2       2   49    male    1     own          little              NaN
3       3   45    male    2    free          little           little
4       4   53    male    2    free          little           little

   Credit amount  Duration              Purpose  Default
0           1169         6             radio/TV        0
1           5951        48             radio/TV        1
2           2096        12            education        0
3           7882        42  furniture/equipment        0
4           4870        24                  car        1
   number  Age     Sex  Job Housing Saving accounts Checking account  \
0       0   67    male    2     own             NaN           little
1       1   22  female    2     own          little         moderate
2       2   49    male    1     own          little              NaN
3       3   45    male    2    free          little           little
4       4   53    male    2    free          little           little

   Credit amount  Duration  Default
0           1169         6        0
1           5951        48        1
2           2096        12        0
3           7882        42        0
4           4870        24        1
   Age     Sex  Job Housing Saving accounts Checking account  Credit amount  \
0   67    male    2     own             NaN           little           1169
1   22  female    2     own          little         moderate           5951
2   49    male    1     own          little              NaN           2096
3   45    male    2    free          little           little           7882
4   53    male    2    free          little           little           4870

   Duration  Default
0         6        0
1        48        1
2        12        0
3        42        0
4        24        1
Age                  int64
Sex                 object
Job                  int64
Housing             object
Saving accounts     object
Checking account    object
Credit amount        int64
Duration             int64
Default              int64
dtype: object
['male' 'female']
['own' 'free' 'rent']
[nan 'little' 'quite rich' 'rich' 'moderate']
   Age     Sex  Job Housing Saving accounts Checking account  Credit amount  \
0   67    male    2     own             NaN           little           1169
1   22  female    2     own          little         moderate           5951
2   49    male    1     own          little              NaN           2096
3   45    male    2    free          little           little           7882
4   53    male    2    free          little           little           4870

   Duration  Default
0         6        0
1        48        1
2        12        0
3        42        0
4        24        1
[nan 'little' 'quite_rich' 'rich' 'moderate']
   Age     Sex  Job Housing Saving accounts Checking account  Credit amount  \
0   67    male    2     own             NaN           little           1169
1   22  female    2     own          little         moderate           5951
2   49    male    1     own          little              NaN           2096
3   45    male    2    free          little           little           7882
4   53    male    2    free          little           little           4870

   Duration
0         6
1        48
2        12
3        42
4        24
0    0
1    1
2    0
3    0
4    1
Name: Default, dtype: int64
Age                  int64
Sex                 object
Job                  int64
Housing             object
Saving accounts     object
Checking account    object
Credit amount        int64
Duration             int64
dtype: object
   Age  Job Housing Saving accounts Checking account  Credit amount  Duration  \
0   67    2     own             NaN           little           1169         6
1   22    2     own          little         moderate           5951        48
2   49    1     own          little              NaN           2096        12
3   45    2    free          little           little           7882        42
4   53    2    free          little           little           4870        24

   Sex_female  Sex_male
0           0         1
1           1         0
2           0         1
3           0         1
4           0         1
   Age  Job  Credit amount  Duration  Sex_female  Sex_male  Housing_free  \
0   67    2           1169         6           0         1             0
1   22    2           5951        48           1         0             0
2   49    1           2096        12           0         1             0
3   45    2           7882        42           0         1             1
4   53    2           4870        24           0         1             1

   Housing_own  Housing_rent  Saving accounts_little  \
0            1             0                       0
1            1             0                       1
2            1             0                       1
3            0             0                       1
4            0             0                       1

   Saving accounts_moderate  Saving accounts_quite_rich  Saving accounts_rich  \
0                         0                           0                     0
1                         0                           0                     0
2                         0                           0                     0
3                         0                           0                     0
4                         0                           0                     0

   Checking account_little  Checking account_moderate  Checking account_rich
0                        1                          0                      0
1                        0                          1                      0
2                        0                          0                      0
3                        1                          0                      0
4                        1                          0                      0
[0 1]
0.3
0.3
0.3
[0]	validation_0-aucpr:0.57051
[1]	validation_0-aucpr:0.57840
[2]	validation_0-aucpr:0.58978
[3]	validation_0-aucpr:0.54545
[4]	validation_0-aucpr:0.58128
[5]	validation_0-aucpr:0.55663
[6]	validation_0-aucpr:0.56723
[7]	validation_0-aucpr:0.56452
[8]	validation_0-aucpr:0.58035
[9]	validation_0-aucpr:0.57406
[10]	validation_0-aucpr:0.57924
[11]	validation_0-aucpr:0.57970
[12]	validation_0-aucpr:0.58447
(750,)
(250,)
C:\Users\yorkuniversity\AppData\Roaming\Python\Python39\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\yorkuniversity\AppData\Roaming\Python\Python39\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
{'gamma': 0.25, 'learning_rate': 0.1, 'max_depth': 3, 'reg_lambda': 10.0, 'scale_pos_weight': 3}
{'gamma': 0.25, 'learning_rate': 0.1, 'max_depth': 2, 'reg_lambda': 20, 'scale_pos_weight': 3}
[01:09:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:576:
Parameters: { "learn_rate" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being used
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[0]	validation_0-aucpr:0.51926
[1]	validation_0-aucpr:0.53845
[2]	validation_0-aucpr:0.55706
[3]	validation_0-aucpr:0.60705
[4]	validation_0-aucpr:0.59142
[5]	validation_0-aucpr:0.64115
[6]	validation_0-aucpr:0.63039
[7]	validation_0-aucpr:0.63246
[8]	validation_0-aucpr:0.65629
[9]	validation_0-aucpr:0.66024
[10]	validation_0-aucpr:0.66951
[11]	validation_0-aucpr:0.66155
[12]	validation_0-aucpr:0.66247
[13]	validation_0-aucpr:0.65216
[14]	validation_0-aucpr:0.66287
[15]	validation_0-aucpr:0.63634
[16]	validation_0-aucpr:0.64834
[17]	validation_0-aucpr:0.65134
[18]	validation_0-aucpr:0.65217
[19]	validation_0-aucpr:0.65029
C:\Users\yorkuniversity\AppData\Roaming\Python\Python39\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[01:09:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:576:
Parameters: { "learn_rate" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being used
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[01:09:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
weight: {'Duration': 1.0, 'Checking account_little': 1.0, 'Checking account_moderate': 1.0}
gain: {'Duration': 11.417625427246094, 'Checking account_little': 53.7900505065918, 'Checking account_moderate': 57.282169342041016}
cover: {'Duration': 90.5, 'Checking account_little': 267.0, 'Checking account_moderate': 176.5}
total_gain: {'Duration': 11.417625427246094, 'Checking account_little': 53.7900505065918, 'Checking account_moderate': 57.282169342041016}
total_cover: {'Duration': 90.5, 'Checking account_little': 267.0, 'Checking account_moderate': 176.5}
C:\Users\yorkuniversity\AppData\Roaming\Python\Python39\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
Error: Could not open "xgboost_tree_risk_customer.pdf" for writing : Permission denied
---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
C:\Users\YORKUN~1\AppData\Local\Temp/ipykernel_17304/1692175857.py in <module>
    399                             leaf_node_params=leaf_params)
    400
--> 401 graph_data.view(filename='xgboost_tree_risk_customer')
    402
    403 #interpreting the tree

~\anaconda3\lib\site-packages\graphviz\files.py in view(self, filename, directory, cleanup, quiet, quiet_view)
    280             to retrieve the application's exit status.
    281         """
--> 282         return self.render(filename=filename, directory=directory,
    283                            view=True, cleanup=cleanup,
    284                            quiet=quiet, quiet_view=quiet_view)

~\anaconda3\lib\site-packages\graphviz\files.py in render(self, filename, directory, view, cleanup, format, renderer, formatter, quiet, quiet_view)
    241             format = self._format
    242
--> 243         rendered = backend.render(self._engine, format, filepath,
    244                                   renderer=renderer, formatter=formatter,
    245                                   quiet=quiet)

~\anaconda3\lib\site-packages\graphviz\backend.py in render(***failed resolving arguments***)
    223         cwd = None
    224
--> 225     run(cmd, capture_output=True, cwd=cwd, check=True, quiet=quiet)
    226     return rendered
    227

~\anaconda3\lib\site-packages\graphviz\backend.py in run(cmd, input, capture_output, check, encoding, quiet, **kwargs)
    183
    184     if check and proc.returncode:
--> 185         raise CalledProcessError(proc.returncode, cmd,
    186                                  output=out, stderr=err)
    187

CalledProcessError: Command '['dot.bat', '-Kdot', '-Tpdf', '-O', 'xgboost_tree_risk_customer']' returned non-zero exit status 1. [stderr: b'Error: Could not open "xgboost_tree_risk_customer.pdf" for writing : Permission denied\r\n']
```





In [ ]:

In [ ]: