# LABORATORIO 18

In [19]:
```
#Importando las librerías necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import model_evaluation_utils as meu

from graphviz import Source
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export_text, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import confusion_matrix, auc, roc_curve
from sklearn.preprocessing import label_binarize
from mlxtend.plotting import plot_decision_regions

#Just in Case
import warnings
warnings.filterwarnings('ignore')

import h2o
from h2o.estimators import H2ORandomForestEstimator
```

In [2]:
```
os.chdir('D:\Social Data Consulting\Python for Data Science\data')
```

In [4]:
```
miArchivo="bank-full.csv"
df_bank=pd.read_csv(miArchivo,sep=';')
df_bank.head()
```

Out[4]:

| | age | job | marital | education | default | balance | housing | loan | contact | day | month | duration | campaign | pd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | management | married | tertiary | no | 2143 | yes | no | unknown | 5 | may | 261 | 1 | |
| **1** | 44 | technician | single | secondary | no | 29 | yes | no | unknown | 5 | may | 151 | 1 | |
| **2** | 33 | entrepreneur | married | secondary | no | 2 | yes | yes | unknown | 5 | may | 76 | 1 | |
| **3** | 47 | blue-collar | married | unknown | no | 1506 | yes | no | unknown | 5 | may | 92 | 1 | |
| **4** | 33 | unknown | single | unknown | no | 1 | no | no | unknown | 5 | may | 198 | 1 | |

In [5]:
```
df_bank.shape
```

Out[5]:
```
(45211, 17)
```

In [7]:
```
columnas=df_bank.columns.to_list()
columnas
```

Out[7]:
```
['age',
 'job',
 'marital',
 'education',
 'default',
 'balance',
 'housing',
 'loan',
 'contact',
 'day',
 'month',
 'duration',
 'campaign',
 'pdays',
 'previous',
 'poutcome',
 'y']
```

In [8]:
```
predictores=columnas[0:len(columnas)-1]
target=columnas[len(columnas)-1]
print(predictores)
print(target)
```

```
['age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'da
y', 'month', 'duration', 'campaign', 'pdays', 'previous', 'poutcome']
y
```

### *1. Dividir la data en 70% de entrenamiento y lo restante para la data de testeo.*

In [16]:
```
X=df_bank.iloc[:,0:df_bank.shape[1]-1].values
y=df_bank.iloc[:,df_bank.shape[1]-1].values
```

In [18]:
```
xtrain,xtest,ytrain,ytest=train_test_split(X,
                                           y,
                                          test_size=0.3,
                                          stratify=y,
                                          random_state=2020)
```

In [29]:
```
df_xtrain=pd.DataFrame(xtrain,columns=predictores)
df_ytrain=pd.DataFrame(ytrain,columns=['y'])

df_xtest=pd.DataFrame(xtest,columns=predictores)
df_ytest=pd.DataFrame(ytest,columns=['y'])

df_train=pd.concat([df_xtrain,df_ytrain],axis=1)
df_test=pd.concat([df_xtest,df_ytest],axis=1)
```

### *2. Entrenar el modelo con los siguientes parametros 60 arboles,3 de profundidad y 8 folders para la validacion cruzada.*

In [33]:
```
h2o.init()
```

```
Checking whether there is an H2O instance running at http://localhost:54321 (http://localhost:54
321) . connected.
Warning: Your H2O cluster version is too old (4 months and 1 day)! Please download and install t
he latest version from http://h2o.ai/download/ (http://h2o.ai/download/)
```

| | |
|---|---|
| H2O_cluster_uptime: | 1 min 08 secs |
| H2O_cluster_timezone: | America/Bogota |
| H2O_data_parsing_timezone: | UTC |
| H2O_cluster_version: | 3.32.0.4 |
| H2O_cluster_version_age: | 4 months and 1 day !!! |
| H2O_cluster_name: | H2O_from_python_Diego_LeÃ³n_lu5goh |
| H2O_cluster_total_nodes: | 1 |
| H2O_cluster_free_memory: | 1.973 Gb |
| H2O_cluster_total_cores: | 8 |
| H2O_cluster_allowed_cores: | 8 |
| H2O_cluster_status: | locked, healthy |
| H2O_connection_url: | http://localhost:54321 |
| H2O_connection_proxy: | {"http": null, "https": null} |
| H2O_internal_security: | False |
| H2O_API_Extensions: | Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4 |
| Python_version: | 3.8.3 final |

In [34]:
```
datah20_train=h2o.H2OFrame(df_train)
```

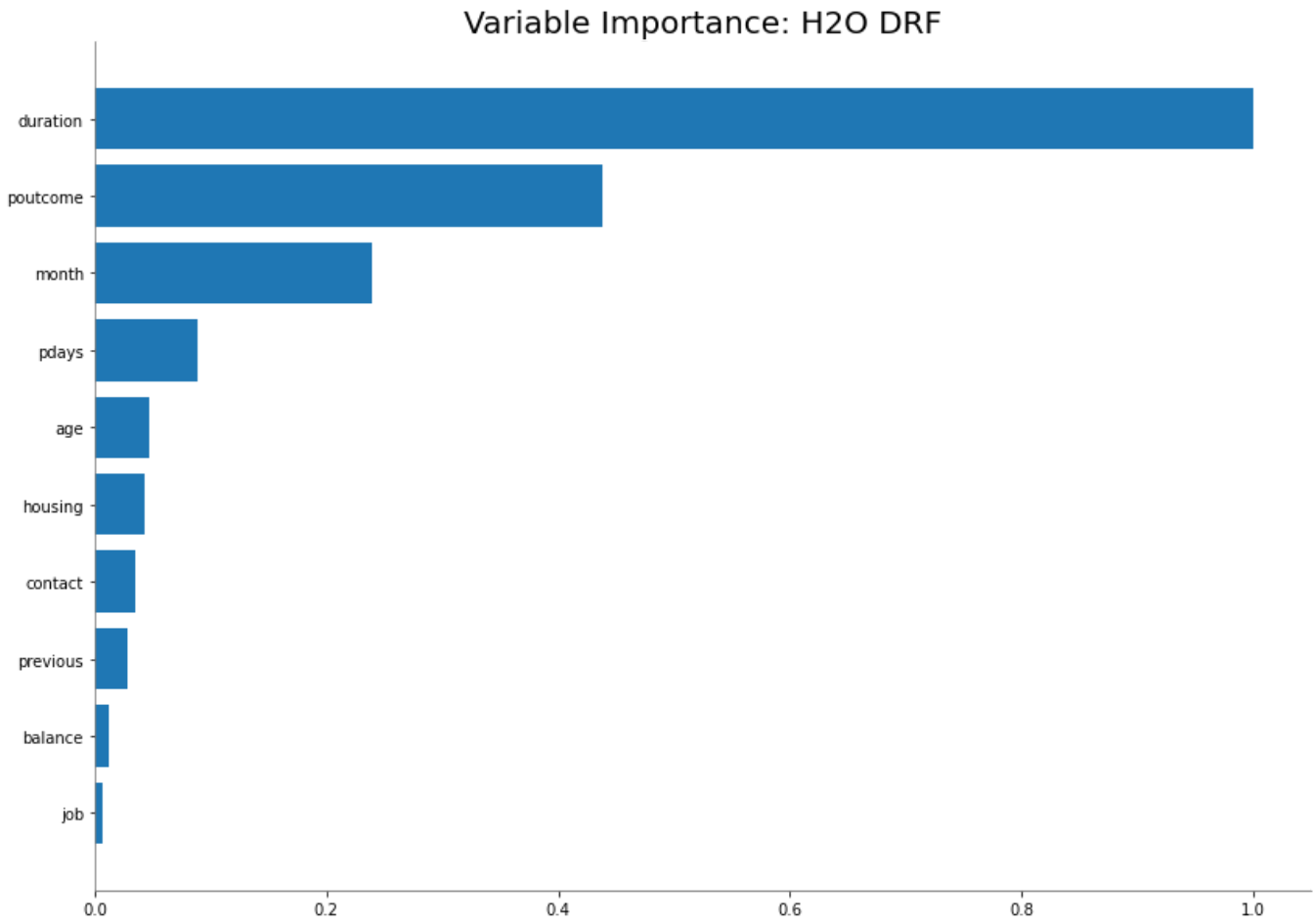
```
Parse progress: |[redacted]| 100%
```

In [35]:
```
#Instanciamos un objeto de clase H2ORandomForestEstimator
m=h2o.estimators.H2ORandomForestEstimator(nfolds=8, #particiones para CV
                                          max_depth=3, #maxima profundidad
                                          ntrees=60, #número de arboles
                                          seed=2020) #semilla
```

In [36]:
```
#Hacer que el modelo aprenda de los datos
m.train(x=predictores,
       y=target,
       training_frame=datah20_train)
```

```
drf Model Build progress: |[redacted]| 100%
```

In [37]:
```
#Identificando a las variables más importantes para Random Forest
m.varimp_plot()
```




***3. Encontrar el AUC del modelo de la validacion cruzada para los datos de entrenamiento.***

In [38]:
```
#Evaluar la performance del modelo con los datos de entrenamiento
performance_train= m.model_performance(datah20_train)
performance_train
```

```
ModelMetricsBinomial: drf
** Reported on test data. **

MSE: 0.07684753744374091
RMSE: 0.2772138839303344
LogLoss: 0.26268415169621
Mean Per-Class Error: 0.17503263095226695
AUC: 0.9002002080377263
AUCPR: 0.5599317373296461
Gini: 0.8004004160754525

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.1505349286292729:
```

| | | no | yes | Error | Rate |
|---|---|---|---|---|---|
| **0** | no | 24634.0 | 3311.0 | 0.1185 | (3311.0/27945.0) |
| **1** | yes | 979.0 | 2723.0 | 0.2645 | (979.0/3702.0) |
| **2** | Total | 25613.0 | 6034.0 | 0.1356 | (4290.0/31647.0) |

```
Maximum Metrics: Maximum metrics at their respective thresholds
```

| | metric | threshold | value | idx |
|---|---|---|---|---|
| **0** | max f1 | 0.150535 | 0.559367 | 245.0 |
| **1** | max f2 | 0.107847 | 0.666531 | 276.0 |
| **2** | max f0point5 | 0.336490 | 0.561638 | 120.0 |
| **3** | max accuracy | 0.336490 | 0.902392 | 120.0 |
| **4** | max precision | 0.618973 | 1.000000 | 0.0 |
| **5** | max recall | 0.046329 | 1.000000 | 394.0 |
| **6** | max specificity | 0.618973 | 1.000000 | 0.0 |
| **7** | max absolute_mcc | 0.147361 | 0.505410 | 247.0 |
| **8** | max min_per_class_accuracy | 0.099409 | 0.819827 | 285.0 |
| **9** | max mean_per_class_accuracy | 0.084928 | 0.824967 | 305.0 |
| **10** | max tns | 0.618973 | 27945.000000 | 0.0 |
| **11** | max fns | 0.618973 | 3701.000000 | 0.0 |
| **12** | max fps | 0.043269 | 27945.000000 | 399.0 |
| **13** | max tps | 0.046329 | 3702.000000 | 394.0 |
| **14** | max tnr | 0.618973 | 1.000000 | 0.0 |
| **15** | max fnr | 0.618973 | 0.999730 | 0.0 |
| **16** | max fpr | 0.043269 | 1.000000 | 399.0 |
| **17** | max tpr | 0.046329 | 1.000000 | 394.0 |

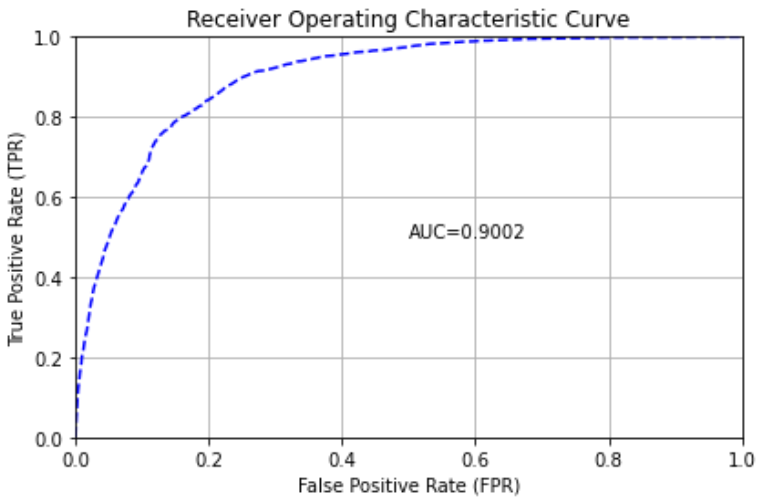
```
Gains/Lift Table: Avg response rate: 11.70 %, avg score: 11.70 %
```

| | group | cumulative_data_fraction | lower_threshold | lift | cumulative_lift | response_rate | score | cumulative_r |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0.010017 | 0.492044 | 6.984521 | 6.984521 | 0.817035 | 0.541393 | |
| **1** | 2 | 0.020002 | 0.433412 | 6.411467 | 6.698447 | 0.750000 | 0.459361 | |

| | group | cumulative_data_fraction | lower_threshold | lift | cumulative_lift | response_rate | score | cumulative_r |
|---|---|---|---|---|---|---|---|---|
| **2** | 3 | 0.030019 | 0.385826 | 5.582223 | 6.325981 | 0.652997 | 0.409420 | |
| **3** | 4 | 0.040004 | 0.366046 | 5.167047 | 6.036705 | 0.604430 | 0.375325 | |
| **4** | 5 | 0.050021 | 0.351212 | 4.503533 | 5.729683 | 0.526814 | 0.358645 | |
| **5** | 6 | 0.100009 | 0.303804 | 4.031146 | 4.880683 | 0.471555 | 0.327336 | |
| **6** | 7 | 0.149998 | 0.246676 | 2.593767 | 4.118538 | 0.303413 | 0.279035 | |
| **7** | 8 | 0.200019 | 0.136685 | 2.705534 | 3.765175 | 0.316488 | 0.193084 | |
| **8** | 9 | 0.300250 | 0.088152 | 1.196592 | 2.907719 | 0.139975 | 0.103097 | |
| **9** | 10 | 0.400006 | 0.077722 | 0.617385 | 2.336538 | 0.072220 | 0.081926 | |
| **10** | 11 | 0.501153 | 0.072120 | 0.288426 | 1.923171 | 0.033739 | 0.074581 | |
| **11** | 12 | 0.600626 | 0.066800 | 0.203668 | 1.638396 | 0.023825 | 0.069348 | |
| **12** | 13 | 0.704964 | 0.061673 | 0.098379 | 1.410465 | 0.011508 | 0.063825 | |
| **13** | 14 | 0.800139 | 0.057572 | 0.034058 | 1.246744 | 0.003984 | 0.059768 | |
| **14** | 15 | 0.900212 | 0.051527 | 0.013496 | 1.109650 | 0.001579 | 0.054765 | |
| **15** | 16 | 1.000000 | 0.042766 | 0.010828 | 1.000000 | 0.001267 | 0.048423 | |

Out[38]:

In [39]:
```
#Podemos obtener la curva ROC y AUC para los Datos de Entrenamiento
performance_train.plot('roc')
```




### *4. Obtener el AUC del modelo con la data de testeo.*

In [40]:
```
datah20_test=h2o.H2OFrame(df_test)
```

```
Parse progress: |[redacted]| 100%
```

In [41]:
```
#Evaluar la performance del modelo con los datos de entrenamiento
performance_test= m.model_performance(datah20_test)
performance_test
```

```
ModelMetricsBinomial: drf
** Reported on test data. **

MSE: 0.07688627247876095
RMSE: 0.2772837400187053
LogLoss: 0.26243678330722836
Mean Per-Class Error: 0.17040776905999044
AUC: 0.9027133185696866
AUCPR: 0.5620574210835447
Gini: 0.8054266371393732

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.1502812096405597:
```

| | | no | yes | Error | Rate |
|---|---|---|---|---|---|
| **0** | no | 10610.0 | 1367.0 | 0.1141 | (1367.0/11977.0) |
| **1** | yes | 424.0 | 1163.0 | 0.2672 | (424.0/1587.0) |
| **2** | Total | 11034.0 | 2530.0 | 0.132 | (1791.0/13564.0) |

```
Maximum Metrics: Maximum metrics at their respective thresholds
```

| | metric | threshold | value | idx |
|---|---|---|---|---|
| **0** | max f1 | 0.150281 | 0.564974 | 237.0 |
| **1** | max f2 | 0.094045 | 0.670413 | 287.0 |
| **2** | max f0point5 | 0.327491 | 0.558024 | 116.0 |
| **3** | max accuracy | 0.347943 | 0.902094 | 99.0 |
| **4** | max precision | 0.606198 | 1.000000 | 0.0 |
| **5** | max recall | 0.051137 | 1.000000 | 382.0 |
| **6** | max specificity | 0.606198 | 1.000000 | 0.0 |
| **7** | max absolute_mcc | 0.150281 | 0.510519 | 237.0 |
| **8** | max min_per_class_accuracy | 0.097318 | 0.820656 | 283.0 |
| **9** | max mean_per_class_accuracy | 0.083994 | 0.829592 | 304.0 |
| **10** | max tns | 0.606198 | 11977.000000 | 0.0 |
| **11** | max fns | 0.606198 | 1586.000000 | 0.0 |
| **12** | max fps | 0.043056 | 11977.000000 | 399.0 |
| **13** | max tps | 0.051137 | 1587.000000 | 382.0 |
| **14** | max tnr | 0.606198 | 1.000000 | 0.0 |
| **15** | max fnr | 0.606198 | 0.999370 | 0.0 |
| **16** | max fpr | 0.043056 | 1.000000 | 399.0 |
| **17** | max tpr | 0.051137 | 1.000000 | 382.0 |

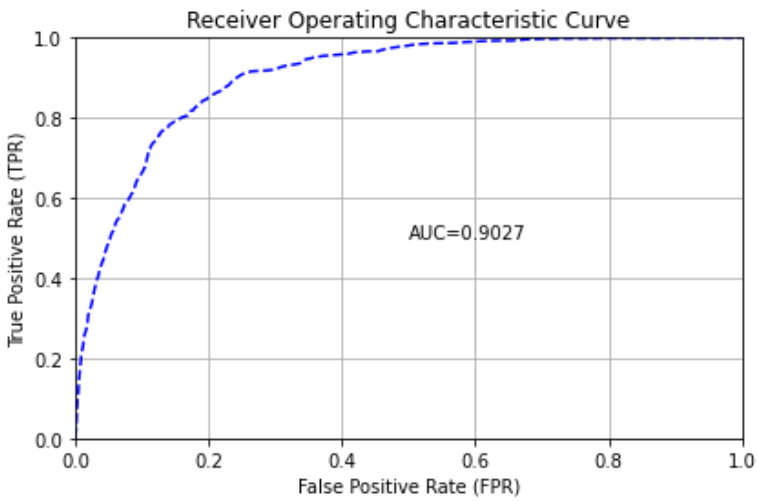
```
Gains/Lift Table: Avg response rate: 11.70 %, avg score: 11.57 %
```

| | group | cumulative_data_fraction | lower_threshold | lift | cumulative_lift | response_rate | score | cumulative_r |
|---|---|---|---|---|---|---|---|---|

| | group | cumulative_data_fraction | lower_threshold | lift | cumulative_lift | response_rate | score | cumulative_r |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0.010027 | 0.477801 | 6.975815 | 6.975815 | 0.816176 | 0.530204 | |
| **1** | 2 | 0.020053 | 0.429942 | 6.535898 | 6.755856 | 0.764706 | 0.452434 | |
| **2** | 3 | 0.030006 | 0.382797 | 6.014516 | 6.509957 | 0.703704 | 0.405548 | |
| **3** | 4 | 0.040032 | 0.366633 | 5.278995 | 6.201650 | 0.617647 | 0.373852 | |
| **4** | 5 | 0.050059 | 0.351145 | 4.399162 | 5.840621 | 0.514706 | 0.359904 | |
| **5** | 6 | 0.100044 | 0.301381 | 3.882682 | 4.862373 | 0.454277 | 0.325779 | |
| **6** | 7 | 0.150029 | 0.243217 | 2.735526 | 4.153773 | 0.320059 | 0.274564 | |
| **7** | 8 | 0.200015 | 0.129436 | 2.710314 | 3.793041 | 0.317109 | 0.183419 | |
| **8** | 9 | 0.300059 | 0.087109 | 1.158908 | 2.914781 | 0.135593 | 0.101435 | |
| **9** | 10 | 0.400029 | 0.077706 | 0.592487 | 2.334421 | 0.069322 | 0.081552 | |
| **10** | 11 | 0.503318 | 0.071546 | 0.317231 | 1.920465 | 0.037116 | 0.074359 | |
| **11** | 12 | 0.599971 | 0.066491 | 0.208621 | 1.644693 | 0.024409 | 0.069085 | |
| **12** | 13 | 0.700310 | 0.061673 | 0.062799 | 1.418042 | 0.007348 | 0.063695 | |
| **13** | 14 | 0.802713 | 0.057235 | 0.055380 | 1.244205 | 0.006479 | 0.059589 | |
| **14** | 15 | 0.899956 | 0.051390 | 0.006480 | 1.110466 | 0.000758 | 0.054574 | |
| **15** | 16 | 1.000000 | 0.042766 | 0.006298 | 1.000000 | 0.000737 | 0.048505 | |

Out[41]:

In [42]:
```
#Podemos obtener la curva ROC y AUC para los Datos de Testeo
performance_test.plot('roc')
```




In [ ]: