2- Calcule o desempenho do modelo de classificação utilizando pelo menos três métricas;

Importação das libs

In [65]:
```
import pandas as pd
import sweetviz as sv
import numpy as np
import sklearn as sk
import matplotlib.pyplot as plt
```

Chamando o arquivo do excel

In [66]:
```
df = pd.read_excel('teste_smarkio_Lbs.xls','Análise_ML')
```

Consultando valores nulo dentro do dataframe

In [67]:
```
df.isnull().sum()
```

Out[67]:
```
Pred_class          0
probabilidade       0
status              0
True_class        462
dtype: int64
```

Tratando os dados da coluna True_class(Todos que o true_class for NaN, considerar o valor da coluna Pred_class)

In [68]:
```
values = {'True_class': df['Pred_class']}
```

In [69]:
```
df.fillna(value=values,inplace = True)
```

Verificando quantas linhas tem dado nulo(depois de rodar o comando para tratar os dados)

In [70]:
```
df.isnull().sum()
```

Out[70]:
```
Pred_class      0
probabilidade   0
status          0
True_class      0
dtype: int64
```

Drop na coluna status, pois não é possível utilizar no modelo.

In [71]:
```
drop_status = df.drop(columns=['status'])
status = df['status']
```

Train and teste split(separando a base para considerar o que é treino e o que e teste)

In [72]:
```
from sklearn.model_selection import train_test_split
```

In [73]:
```
x_train, x_test, y_train, y_test = train_test_split(drop_status,status,test_size=0.5,
```

Regressão Logística:

In [74]:
```
from sklearn.linear_model import LogisticRegression
```

In [75]:
```
lr = LogisticRegression(random_state=663, solver = 'liblinear')
```

In [76]:
```
lr.fit(x_train, y_train)
```

Out[76]:
```
LogisticRegression(random_state=663, solver='liblinear')
```

In [77]:
```
x_train.shape
```

Out[77]:
```
(321, 3)
```

In [78]:
```
x_test.shape
```

Out[78]:
```
(322, 3)
```

Acurácia do modelo:

In [79]:
```
from sklearn.metrics import accuracy_score
```

In [80]:
```
accuracy_lr_train = round(accuracy_score(y_train, lr.predict(x_train))*100,3)
```

In [81]:
```
accuracy_lr_test = round(accuracy_score(y_test, lr.predict(x_test))*100,3)
```

In [82]:
```
print('Acuracia do treino', accuracy_lr_train)
print('Acuracia do teste:', accuracy_lr_test)
```

```
Acuracia do treino 95.016
Acuracia do teste: 91.615
```

In [83]:
```
from sklearn.metrics import classification_report
```

Classificação Treino

In [84]:
```
print(classification_report(y_train,  lr.predict(x_train)))
```

```
              precision    recall  f1-score   support

    approved       0.95      1.00      0.97       305
    revision       0.00      0.00      0.00        16

    accuracy                           0.95       321
   macro avg       0.48      0.50      0.49       321
weighted avg       0.90      0.95      0.93       321
```

Classificação teste:

In [85]:
```
print(classification_report(y_test,  lr.predict(x_test)))
```

```
              precision    recall  f1-score   support

    approved       0.92      1.00      0.96       295
    revision       0.00      0.00      0.00        27

    accuracy                           0.92       322
   macro avg       0.46      0.50      0.48       322
weighted avg       0.84      0.92      0.88       322
```

DecisionTree

In [86]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [87]:
```
dt2 = DecisionTreeClassifier(criterion='gini',random_state = 123, max_depth=15, max_l
```

dt2.fit(x_train, y_train)

In [89]:
```
accuracy_dtc_train = round(accuracy_score(y_train, dt2.predict(x_train)) *100, 2)
```

In [90]:
```
accuracy_dtc_test = round(accuracy_score(y_test, dt2.predict(x_test)) *100, 2)
```

In [91]:
```
print('Acuracia de treino da arvore de decisao',accuracy_dtc_train)
print('Acuracia de teste da arvore de decisao', accuracy_dtc_test)
```

```
Acuracia de treino da arvore de decisao 98.13
Acuracia de teste da arvore de decisao 90.99
```

Accuracy Treino Decision Tree

In [92]:
```
print(classification_report(y_train,  dt2.predict(x_train)))
```

```
              precision    recall  f1-score   support

    approved       0.98      1.00      0.99       305
    revision       1.00      0.62      0.77        16

    accuracy                           0.98       321
   macro avg       0.99      0.81      0.88       321
weighted avg       0.98      0.98      0.98       321
```

Accuracy Teste Decision Tree

In [94]:
```
print(classification_report(y_test, dt2.predict(x_test)))
```

```
              precision    recall  f1-score   support

    approved       0.92      0.99      0.95       295
    revision       0.25      0.04      0.06        27

    accuracy                           0.91       322
   macro avg       0.58      0.51      0.51       322
weighted avg       0.86      0.91      0.88       322
```

Random Florest

In [95]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [96]:
```
rndforest = RandomForestClassifier(criterion='entropy',random_state=123,min_samples_le
```

In [97]:
```
rndforest.fit(x_train, y_train)
```

Out[97]:
```
RandomForestClassifier(bootstrap='bool', criterion='entropy',
                       min_samples_leaf=10, n_estimators=700, random_state=123)
```

Treino

In [98]:
```
y_trainrnd = rndforest.predict(x_train)
y_score_trainrnd = rndforest.predict_proba(x_train)[:,1]
```

Teste

In [99]:
```
y_testrnd = rndforest.predict(x_test)
y_score_trainrnd = rndforest.predict_proba(x_test)[:,1]
```

In [100…
```
accuracy_train_rnd = round(accuracy_score(y_trainrnd, y_train) * 100, 2)
accuracy_test_rnd = round(accuracy_score(y_testrnd, y_test) * 100, 2)
```

In [101…
```
print('Acuracia de treino Random Forest:', accuracy_train_rnd)
print('Acuracia de teste  Random Forest:', accuracy_test_rnd)
```

```
Acuracia de treino Random Forest: 95.02
Acuracia de teste  Random Forest: 91.61
```

Accuracy Treino Random Forest

In [102…
```
print(classification_report(y_train,  rndforest.predict(x_train)))
```

```
              precision    recall  f1-score   support

    approved       0.95      1.00      0.97       305
    revision       0.00      0.00      0.00        16

    accuracy                           0.95       321
   macro avg       0.48      0.50      0.49       321
weighted avg       0.90      0.95      0.93       321
```

Accuracy Teste Random Forest

In [103…
```
print(classification_report(y_test,  rndforest.predict(x_test)))
```

```
              precision    recall  f1-score   support

    approved       0.92      1.00      0.96       295
    revision       0.00      0.00      0.00        27

    accuracy                           0.92       322
   macro avg       0.46      0.50      0.48       322
weighted avg       0.84      0.92      0.88       322
```

In [62]:
```
models = pd.DataFrame ({
    'Modelo': ['Regrsão Logistica',
               'Decision Tree',
               'Random Forest'],
    'Accuracy_train': [accuracy_lr_train,
                       accuracy_dtc_train,
                       accuracy_train_rnd],
    'Accuracy_test': [ accuracy_lr_test,
                      accuracy_dtc_test,
                       accuracy_test_rnd]
})

model_comp = models.sort_values(by= 'Accuracy_test', ascending=False)
model_comp = models.sort_values(by= 'Accuracy_train', ascending=False)
```

In [63]:
```
model_comp
```

Out[63]:

	Modelo	Accuracy_train	Accuracy_test
1	Decision Tree	98.130	90.990
2	Random Forest	95.020	91.610
0	Regrsão Logistica	95.016	91.615