Questoes_3_e_4

February 28, 2021

1 Questões 3 e 4

In [1]:
```
#Módulos utilizados:
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
path = "teste_smarkio_lbs.xls"
df1 = pd.read_excel(path, 'Análise_ML')
```

In [3]:
```
df1.head()
```

Out[3]:

	Pred_class	probabilidade	status	True_class
0	2	0.079892	approved	0.0
1	2	0.379377	approved	74.0
2	2	0.379377	approved	74.0
3	2	0.420930	approved	74.0
4	2	0.607437	approved	NaN

In [4]:
```
#Separar o que é approved para amostra de treino e validacao

df_ap = df1[df1['status']=='approved']
```

In [5]:
```
#Criação da variavel alvo: 1 para previu certo e 0 para previsoes erradas
df_ap['Classe_verdadeira'] = df_ap['True_class']
df_ap['Classe_verdadeira'] = np.where(df_ap['Classe_verdadeira'].notnull(),0,1)
```

In [6]:
```
df_ap.head()
```

Out[6]:

	Pred_class	probabilidade	status	True_class	Classe_verdadeira
0	2	0.079892	approved	0.0	0
1	2	0.379377	approved	74.0	0
2	2	0.379377	approved	74.0	0
3	2	0.420930	approved	74.0	0
4	2	0.607437	approved	NaN	1

```
In [7]: #Distribuição da variavel alvo
        df_ap['Classe_verdadeira'].value_counts()

Out[7]: 1    419
        0    181
        Name: Classe_verdadeira, dtype: int64

In [8]: #Remocao das colunas status e True_class
        df_ap.drop(columns=['status','True_class'],axis='columns',inplace=True)

In [9]: df_ap.head()

Out[9]:    Pred_class  probabilidade  Classe_verdadeira
        0           2       0.079892                  0
        1           2       0.379377                  0
        2           2       0.379377                  0
        3           2       0.420930                  0
        4           2       0.607437                  1

In [10]: df_ap.isnull().any()

Out[10]: Pred_class           False
         probabilidade        False
         Classe_verdadeira    False
         dtype: bool

In [11]: #Separacao da variavel dependente das variaveis explanatorias
         X=df_ap.iloc[:,:-1]
         y=df_ap.iloc[:,-1:]
```

1.1 Treino de um Random Forest

```
In [14]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import cross_validate

         modelo_rf = RandomForestClassifier(n_estimators=200,\
                                            criterion = 'entropy',\
                                            random_state=42)

         #treino com cross validacao
         scores_rf = cross_validate(modelo_rf,X,y,scoring=('accuracy',\
                                                           'precision',\
                                                           'recall',\
                                                           'f1') )

         scores_rf

Out[14]: {'fit_time': array([0.60637951, 0.60039258, 0.51562142, 0.53855944, 0.54953146]),
          'score_time': array([0.0249331 , 0.02393627, 0.0478723 , 0.0578444 , 0.0249331 ]),
```

```
 'test_accuracy': array([0.70833333, 0.71666667, 0.83333333, 0.76666667, 0.75      ])
 'test_f1': array([0.78527607, 0.78481013, 0.88095238, 0.83529412, 0.84210526]),
 'test_precision': array([0.8       , 0.83783784, 0.88095238, 0.8255814 , 0.75471698])
 'test_recall': array([0.77108434, 0.73809524, 0.88095238, 0.8452381 , 0.95238095])}
```

```
In [15]: acuracia_media_rf = scores_rf['test_accuracy'].mean()
         precisao_media_rf = scores_rf['test_precision'].mean()
         recall_media_rf =  scores_rf['test_recall'].mean()
         f1_media_rf = scores_rf['test_f1'].mean()
         print('Médias: \n - Acuracia:{},\n - Precisao:{},\n - Recall:{},\n - F1:{}'.
               format(acuracia_media_rf,precisao_media_rf,recall_media_rf,f1_media_rf))
```

```
Médias:
 - Acuracia:0.755,
 - Precisao:0.8198177190542264,
 - Recall:0.8375502008032129,
 - F1:0.825687592391849
```

1.2 Treino de um XGBoost

```
In [16]: from xgboost import XGBClassifier

         modelo_x = XGBClassifier(n_estimators = 1000,\
                                  learning_rate = 0.05,\
                                  seed=42)

         #treino com cross validacao
         scores_x= cross_validate(modelo_x,X,y,scoring=('accuracy',\
                                                        'precision',\
                                                        'recall',\
                                                        'f1') )
```

```
In [17]: acuracia_media_x = scores_x['test_accuracy'].mean()
         precisao_media_x = scores_x['test_precision'].mean()
         recall_media_x =  scores_x['test_recall'].mean()
         f1_media_x = scores_x['test_f1'].mean()
         print('Médias: \n - Acuracia: {},\n - Precisao: {},\n - Recall :{},\n - F1: {}'.
               format(acuracia_media_x,precisao_media_x,recall_media_x,f1_media_x))
```

```
Médias:
 - Acuracia: 0.7466666666666667,
 - Precisao: 0.8196536553737207,
 - Recall :0.8187894434882386,
 - F1: 0.8175796902122275
```

1.3 Métricas

Considerando: - VP = verdadeiros positivos - FP = falsos posistivos - FN = falsos negativos - VF = verdadeiros falsos

Vamos utilizar as seguintes métricas:

- A **Acurácia**, que mede o numero de predições corretas em relação ao total de predições ((VP+VF)/(VP+FP+FN+VF))
- A **Precisão**, que quantifica o número de verdadeiros positivos dentre todas as previsões positivas (VP/(VP+FP))
- O **Recall**, que quantifica o numero de verdadeiros positivos dentro todos os exemplos positivos do dataset (VP/(VP+FN))

1.4 Previsão

Uma vez que as métricas do Random Forest foram ligeiramente superiores, vamos utilizá-lo para a previsão no set de dados com status de revision

```
In [18]: #Filtrando do set de dados o status de revision
         df_rev = df1[df1['status']=='revision']

In [19]: #tamanho do set de dados
         len(df_rev)

Out[19]: 43

In [20]: #Remocao das colunas status e True_class
         df_rev.drop(columns=['status','True_class'],axis='columns',inplace=True)

In [21]: modelo_escolhido = modelo_rf.fit(X,y)

In [22]: pred_rf = modelo_escolhido.predict(df_rev)

In [23]: df_rev['Predicao correta'] = pred_rf

In [24]: #1 - previu certo e 0 - previu errado
         df_rev

Out[24]:      Pred_class  probabilidade  Predicao correta
         83            2       0.752448                 1
         164           3       0.784920                 1
         191          12       0.574756                 0
         201          17       0.506654                 0
         236          24       0.817525                 1
         237          24       0.909148                 1
         238          24       0.737133                 1
         239          24       0.318306                 1
         242          25       0.509871                 1
         243          25       0.629700                 1
         244          25       0.633426                 1
```

```
264          32       0.621226                0
265          36       0.285545                0
268          39       0.812112                1
269          39       0.812112                1
277          43       0.725794                1
281          43       0.740075                1
319          55       0.740292                1
320          55       0.675269                1
372          60       0.511118                1
373          60       0.543772                1
376          60       0.553846                1
457          77       0.419723                0
459          77       0.606065                1
487          84       0.561842                1
503          86       0.545478                1
546          96       0.340740                1
608          77       0.403734                0
611         114       0.487069                1
612          11       0.320702                0
613          24       0.287126                1
615           2       0.331168                1
616           3       0.399808                1
617           4       0.405327                0
618          22       0.324137                1
623          60       0.421998                1
624          81       0.351401                1
625          96       0.313003                1
627           2       0.334350                0
628           3       0.351031                0
630           4       0.278516                1
631           4       0.301915                1
632         113       0.516733                1
```

In [25]:
```
df_rev['Predicao correta'].value_counts()
```

Out[25]:
```
1    33
0    10
Name: Predicao correta, dtype: int64
```