# E2 - Classificação com Regressão Logística (Análise crédito)

"South German Credit" é uma base com informações de 1000 créditos (700 good and 300 bad) com 20 variáveis preditoras.

Objetivo: desenvolver um modelo de predição do risco de crédito (0 = bom; 1 = ruim)

- Importando as bibliotecas

```
import pandas as pd
import numpy as np
import statsmodels
import seaborn
from matplotlib import pyplot as plt
pd.options.display.max_columns = 100
pd.options.mode.chained_assignment = None  # default='warn'
```

```
/Users/karinseeder/anaconda3/lib/python3.7/site-
packages/pandas/compat/_optional.py:138: UserWarning: Pandas requires
version '2.7.0' or newer of 'numexpr' (version
'2.6.8' currently installed).
  warnings.warn(msg, UserWarning)
```

- Importando a base de dados

```
df = pd.read_csv('base_2sgc.csv')
print(df.shape)
```

```
(1000, 21)
```

- Visualizando uma amostra da base

```
df.head()
```

| | status | duration | credit_history | purpose | amount | savings | employment_duration | installment_rate | p |
|---|---|---|---|---|---|---|---|---|---|
| 0 | no checking account | 18 | all credits at this bank paid back duly | car (used) | 1049 | unknown/no savings account | < 1 yr | < 20 | f c |
| 1 | no checking account | 9 | all credits at this bank paid back duly | others | 2799 | unknown/no savings account | 1 <= ... < 4 yrs | 25 <= ... < 35 | r r |
| 2 | ... < 0 DM | 12 | no credits taken/all credits paid back duly | retraining | 841 | ... < 100 DM | 4 <= ... < 7 yrs | 25 <= ... < 35 | f c |
| 3 | no checking account | 12 | all credits at this bank paid back duly | others | 2122 | unknown/no savings account | 1 <= ... < 4 yrs | 20 <= ... < 25 | r r |
| 4 | no checking account | 12 | all credits at this bank paid back duly | others | 2171 | unknown/no savings account | 1 <= ... < 4 yrs | < 20 | r r |

# 1 - Análise descritiva de variáveis

1.1 Estatísticas descritivas: frequência, tabelas cruzadas, média ($\bar{x}$), desvio padrão ($s$), quartis ($Q1$, $\tilde{x}$ , $Q3$) (2,0)

- Avaliando o tipo das variáveis na base

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   status                  1000 non-null   object
 1   duration                1000 non-null   int64 
 2   credit_history          1000 non-null   object
 3   purpose                 1000 non-null   object
 4   amount                  1000 non-null   int64 
 5   savings                 1000 non-null   object
 6   employment_duration     1000 non-null   object
 7   installment_rate        1000 non-null   object
 8   personal_status_sex     1000 non-null   object
 9   other_debtors           1000 non-null   object
 10  present_residence       1000 non-null   object
 11  property                1000 non-null   object
 12  age                     1000 non-null   int64 
 13  other_installment_plans 1000 non-null   object
 14  housing                 1000 non-null   object
 15  number_credits          1000 non-null   object
 16  job                     1000 non-null   object
 17  people_liable           1000 non-null   object
 18  telephone               1000 non-null   object
 19  foreign_worker          1000 non-null   object
 20  credit_risk             1000 non-null   object
dtypes: int64(3), object(18)
memory usage: 164.2+ KB
```

- Análise descritiva para variáveis numéricas

```
df.describe()
```

| | duration | amount | age |
|---|---|---|---|
| count | 1000.000000 | 1000.00000 | 1000.00000 |
| mean | 20.903000 | 3271.24800 | 35.54200 |
| std | 12.058814 | 2822.75176 | 11.35267 |
| min | 4.000000 | 250.00000 | 19.00000 |
| 25% | 12.000000 | 1365.50000 | 27.00000 |
| 50% | 18.000000 | 2319.50000 | 33.00000 |
| 75% | 24.000000 | 3972.25000 | 42.00000 |
| max | 72.000000 | 18424.00000 | 75.00000 |

- Análise descritiva das variáveis categóricas

```
df.describe(include=object)
```

| | status | credit_history | purpose | savings | employment_duration | installment_rate | personal_sta |
|---|---|---|---|---|---|---|---|
| count | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| unique | 4 | 5 | 10 | 5 | 5 | 4 | 4 |
| top | ... >= 200 DM / salary for at least 1 year | no credits taken/all credits paid back duly | furniture/equipment | unknown/no savings account | 1 <= ... < 4 yrs | < 20 | male : married/wic |

| | status | credit_history | purpose | savings | employment_duration | installment_rate | personal_sta |
|---|---|---|---|---|---|---|---|
| freq | 394 | 530 | 280 | 603 | 339 | 476 | 548 |

```
plt.rcParams["figure.figsize"] = (20,10)
for i, x in enumerate(df.dtypes):
  if x == 'object':
    #print(pd.crosstab(index=df[df.columns[i]], columns='freq', dropna=False))
    print(pd.crosstab(df[df.columns[i]], df['credit_risk']).apply(lambda r: r/r.sum(), axis=0))
    print('')
```

```
credit_risk                                     bad      good
status
... < 0 DM                                 0.350000  0.234286
... >= 200 DM / salary for at least 1 year  0.153333  0.497143
0<= ... < 200 DM                           0.046667  0.070000
no checking account                        0.450000  0.198571

credit_risk                                  bad      good
credit_history
all credits at this bank paid back duly      0.166667  0.347143
critical account/other credits elsewhere     0.093333  0.030000
delay in paying off in the past              0.083333  0.021429
existing credits paid back duly till now     0.093333  0.085714
no credits taken/all credits paid back duly  0.563333  0.515714

credit_risk               bad      good
purpose
business             0.016667  0.010000
car (new)            0.056667  0.122857
car (used)           0.193333  0.175714
domestic appliances  0.026667  0.020000
furniture/equipment  0.206667  0.311429
others               0.296667  0.207143
radio/television     0.013333  0.011429
repairs              0.073333  0.040000
retraining           0.113333  0.090000
vacation             0.003333  0.011429

credit_risk                      bad      good
savings
... <  100 DM               0.113333  0.098571
... >= 1000 DM              0.106667  0.215714
100 <= ... <  500 DM        0.036667  0.074286
500 <= ... < 1000 DM        0.020000  0.060000
unknown/no savings account  0.723333  0.551429

credit_risk               bad      good
employment_duration
1 <= ... < 4 yrs     0.346667  0.335714
4 <= ... < 7 yrs     0.130000  0.192857
< 1 yr               0.233333  0.145714
>= 7 yrs             0.213333  0.270000
unemployed           0.076667  0.055714

credit_risk            bad      good
installment_rate
20 <= ... < 25    0.150000  0.160000
25 <= ... < 35    0.206667  0.241429
< 20              0.530000  0.452857
>= 35             0.113333  0.145714

credit_risk                                 bad      good
personal_status_sex
female : non-single or male : single  0.363333  0.287143
female : single                       0.083333  0.095714
male : divorced/separated             0.066667  0.042857
male : married/widowed                0.486667  0.574286

credit_risk         bad      good
other_debtors
co-applicant   0.060000  0.032857
guarantor      0.033333  0.060000
none           0.906667  0.907143

credit_risk              bad      good
present_residence
1 <= ... < 4 yrs    0.323333  0.301429
4 <= ... < 7 yrs    0.143333  0.151429
< 1 yr              0.120000  0.134286
>= 7 yrs            0.413333  0.412857

credit_risk                                    bad      good
```

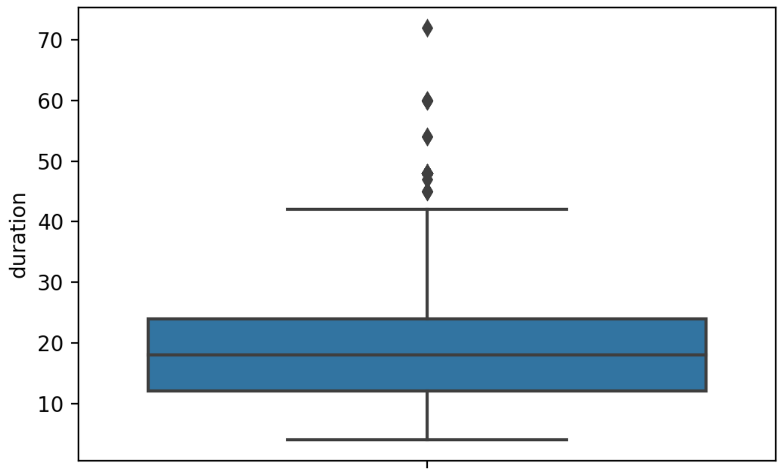
```
property
building soc. savings agr./life insurance  0.340000  0.328571
car or other                               0.236667  0.230000
real estate                                0.223333  0.124286
unknown / no property                      0.200000  0.317143

credit_risk                  bad      good
other_installment_plans
bank                    0.190000  0.117143
none                    0.746667  0.842857
stores                  0.063333  0.040000

credit_risk       bad      good
housing
for free     0.233333  0.155714
own          0.146667  0.090000
rent         0.620000  0.754286

credit_risk          bad      good
number_credits
1               0.666667  0.618571
2-3             0.306667  0.344286
4-5             0.020000  0.031429
>= 6            0.006667  0.005714

credit_risk                                     bad      good
job
manager/self-empl./highly qualif. employee  0.170000  0.138571
skilled employee/official                   0.620000  0.634286
unemployed/unskilled - non-resident         0.023333  0.021429
unskilled - resident                        0.186667  0.205714

credit_risk         bad      good
people_liable
0 to 2         0.846667  0.844286
3 or more      0.153333  0.155714

credit_risk                    bad      good
telephone
no                        0.623333  0.584286
yes (under customer name)  0.376667  0.415714

credit_risk          bad      good
foreign_worker
no              0.986667  0.952857
yes             0.013333  0.047143

credit_risk  bad  good
credit_risk
bad          1.0   0.0
good         0.0   1.0
```

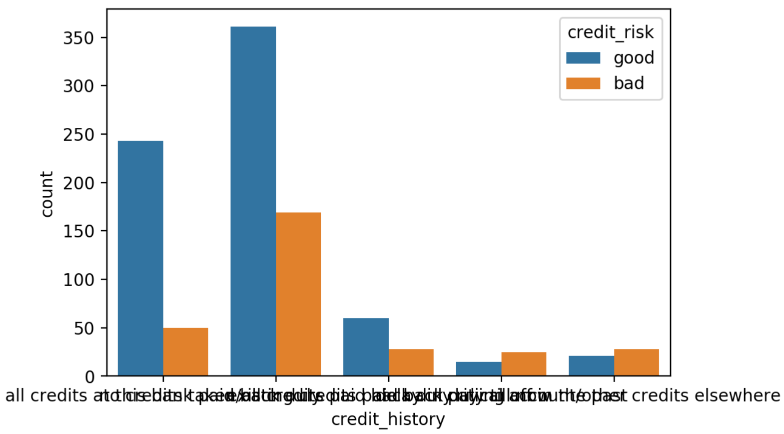
1.2 Gráficos como: Gráficos de colunas, BoxPlot por categorias (2,0)

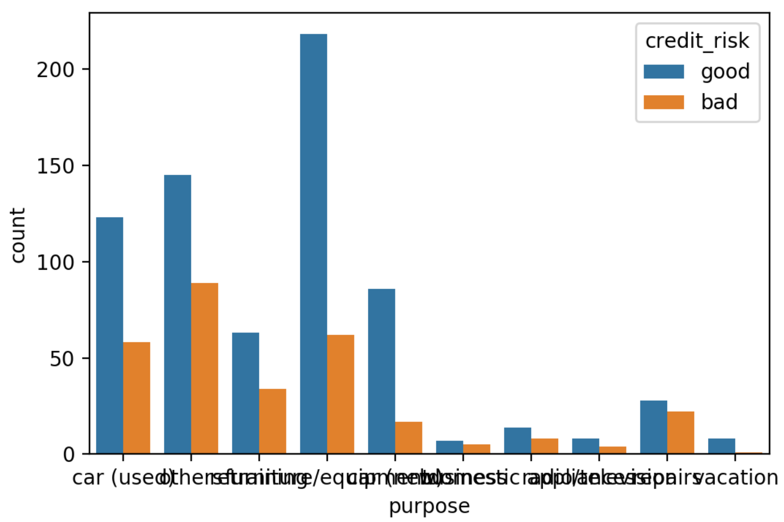
```python
import seaborn as sns

for i, x in enumerate(df.dtypes):
   if x == 'int64' or x == 'float64':
       plt.figure(i)
       sns.boxplot(data = df, y = df[df.columns[i]], hue = df['credit_risk'])
   elif x == 'object':
       plt.figure(i)
       sns.countplot(x=df[df.columns[i]], hue = 'credit_risk', data=df)
```

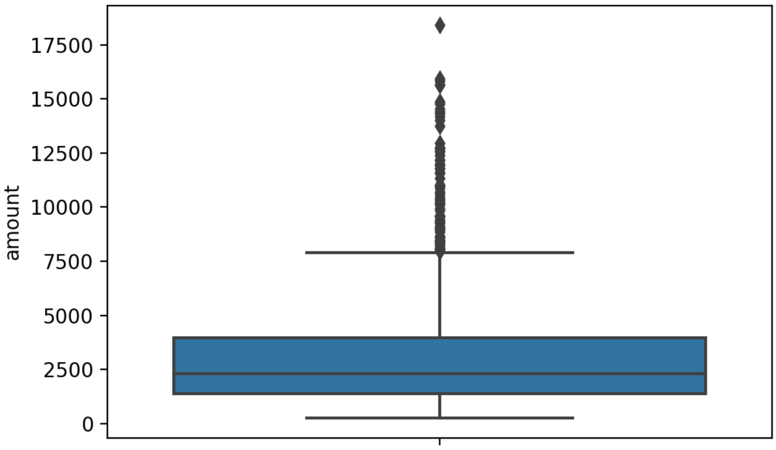
```
/Users/karinseeder/anaconda3/lib/python3.7/site-
packages/matplotlib/pyplot.py:514: RuntimeWarning: More than 20
figures have been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and
may consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  max_open_warning, RuntimeWarning)
```


credit_risk
good
bad
count
no checking account
status

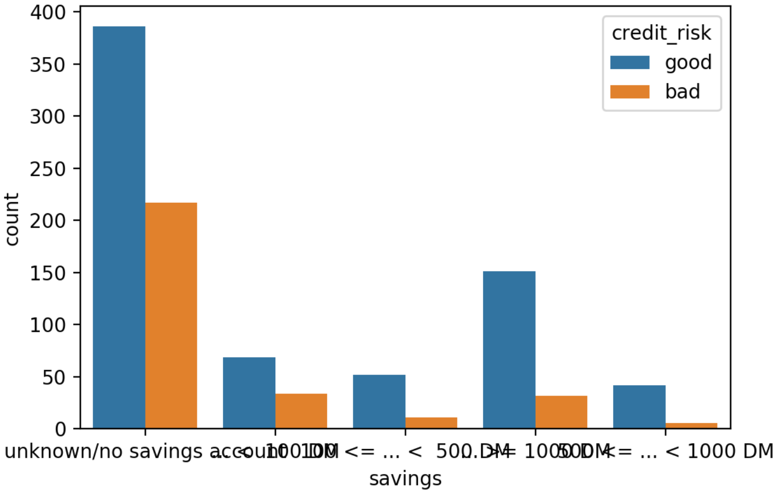

duration


credit_risk
good
bad
count
credit_history

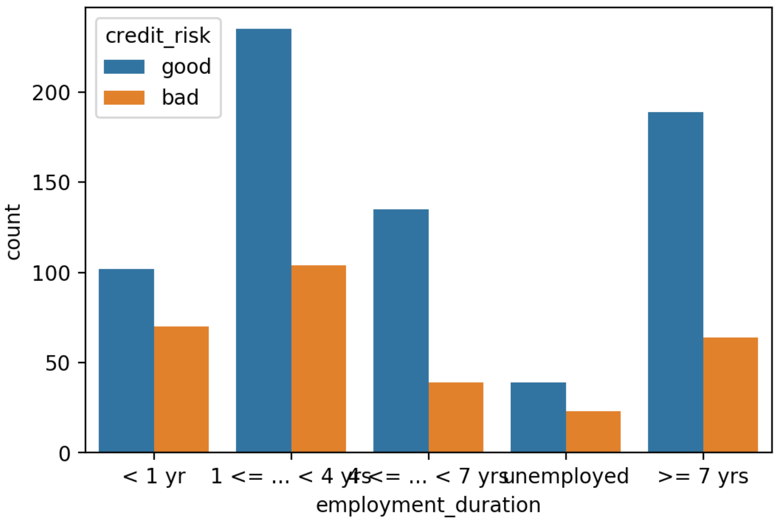

credit_risk
good
bad
count
car (used)
vacation
purpose



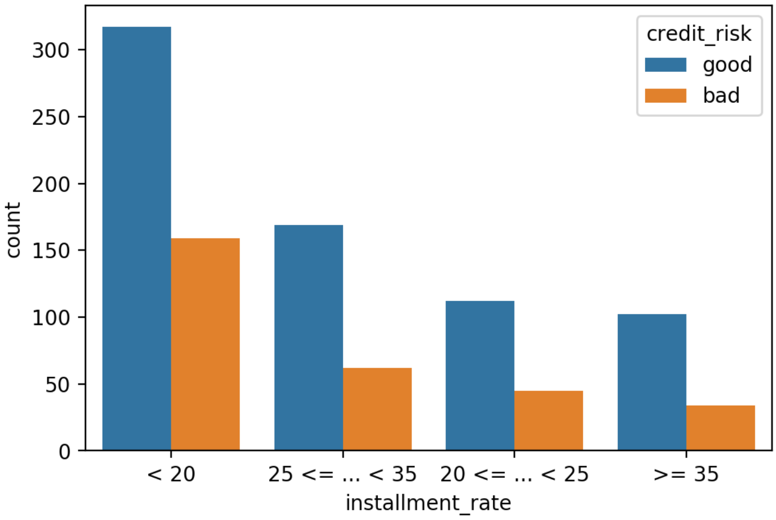
17500
15000
12500
10000
7500
5000
2500
0
amount

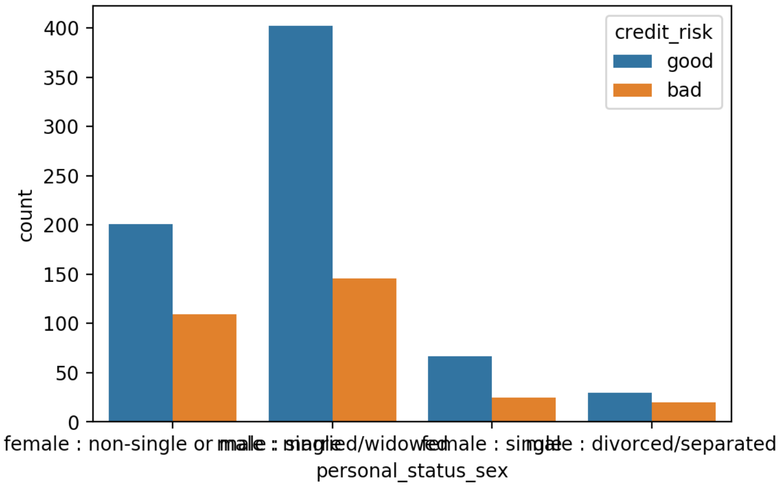



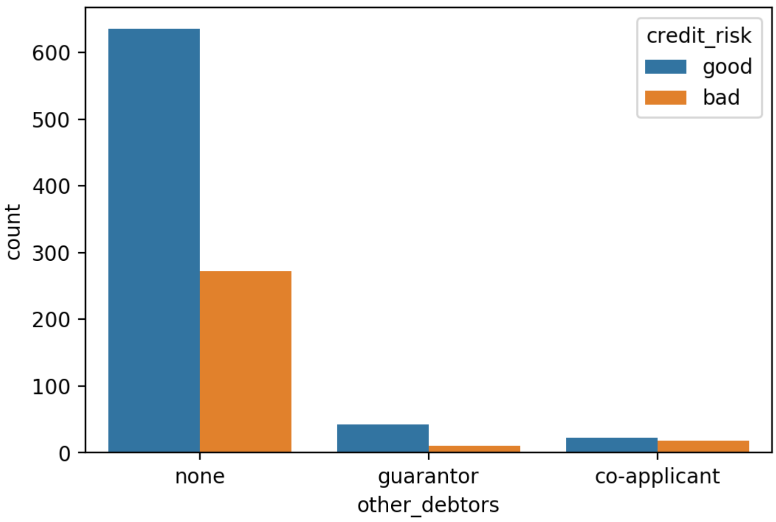
400
350
300
250
200
150
100
50
0
count
credit_risk
good
bad
unknown/no savings account
... < 100 DM
100 DM <= ... < 500 DM
... >= 1000 DM
500 DM <= ... < 1000 DM
savings

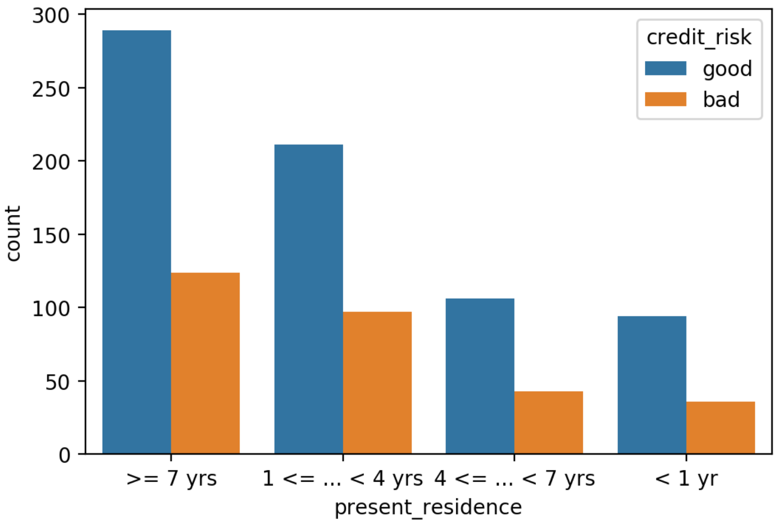



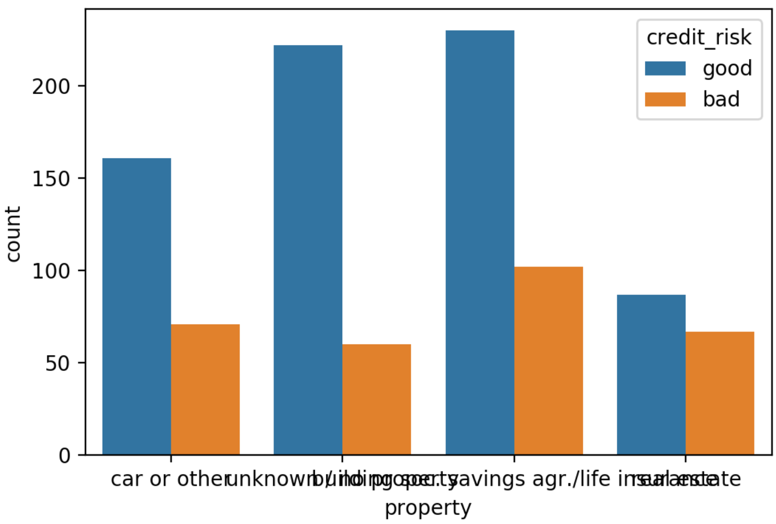
200
150
100
50
0
count
credit_risk
good
bad
< 1 yr
1 <= ... < 4 yrs
4 <= ... < 7 yrs
unemployed
>= 7 yrs
employment_duration

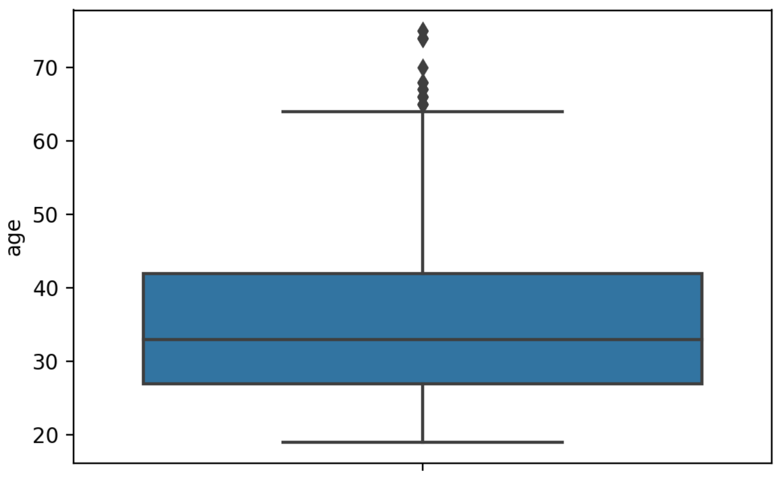

credit_risk
good
bad
300
250
200
150
100
50
0
count
< 20
25 <= ... < 35
20 <= ... < 25
>= 35
installment_rate

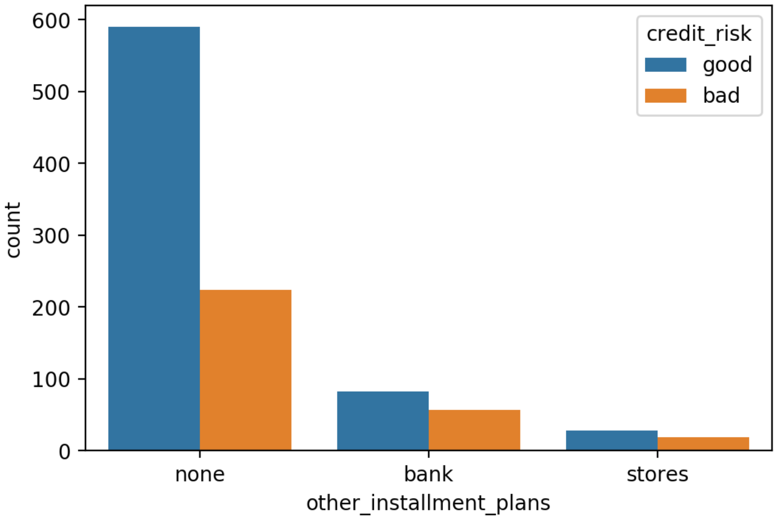

credit_risk
good
bad
400
350
300
250
200
150
100
50
0
count
personal_status_sex

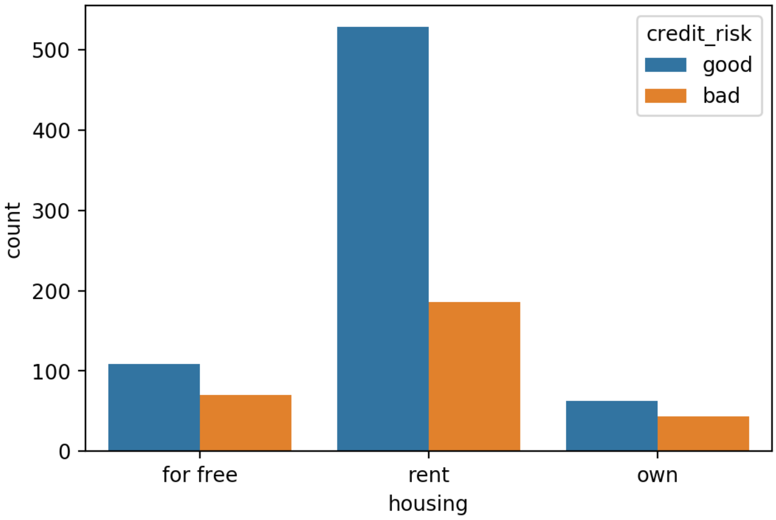

credit_risk
good
bad
600
500
400
300
200
100
0
count
none
guarantor
co-applicant
other_debtors

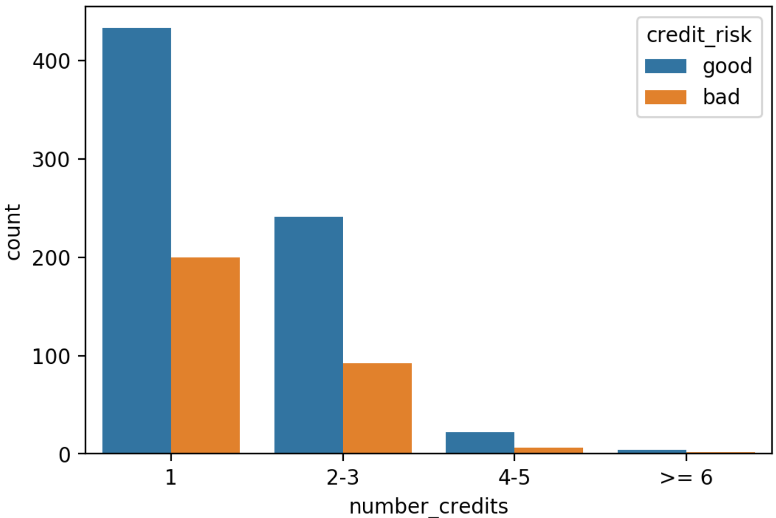

credit_risk
good
bad
count
present_residence
>= 7 yrs
1 <= ... < 4 yrs
4 <= ... < 7 yrs
< 1 yr

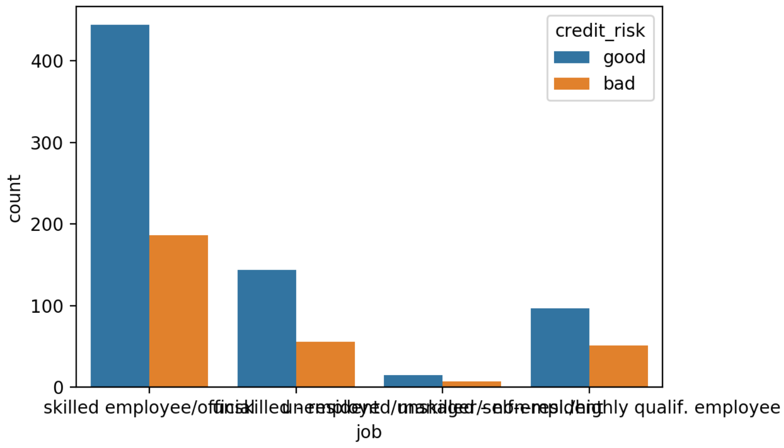

credit_risk
good
bad
count
property
car or other
real estate

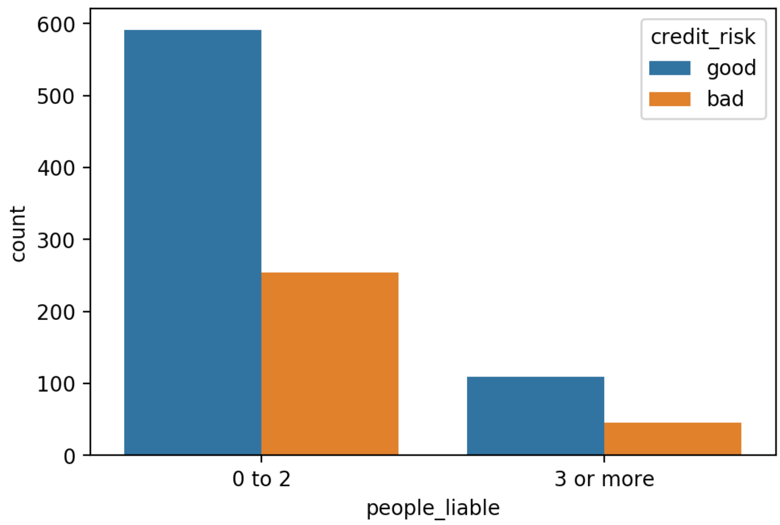

age

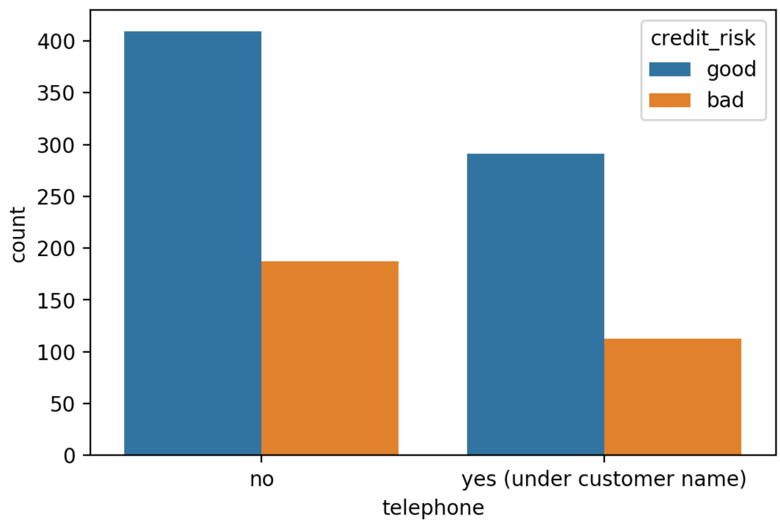

credit_risk
good
bad
count
0
100
200
300
400
500
600
none
bank
stores
other_installment_plans

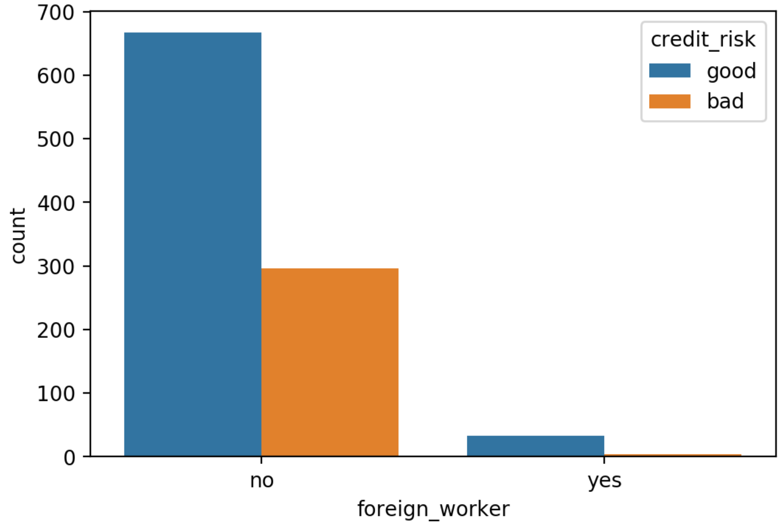

credit_risk
good
bad
count
0
100
200
300
400
500
for free
rent
own
housing

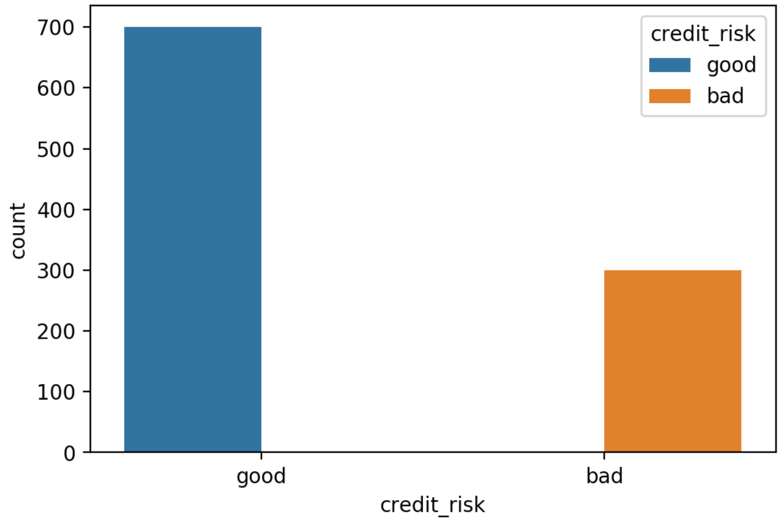

credit_risk
good
bad
count
0
100
200
300
400
1
2-3
4-5
>= 6
number_credits


credit_risk
good
bad
count
job
skilled employee/official
unskilled - resident
unemployed/unskilled - non-resident
manager/self-empl./highly qualif. employee


credit_risk
good
bad
count
0 to 2
3 or more
people_liable


credit_risk
good
bad
count
no
yes (under customer name)
telephone




# 2 - Desenvolvimento de modelo de Classificação utilizando Regressão Logística (binomial).

- Primeiramente realizamos um tratamento nas variaveis

```
cols = df.columns
num_data = list(df._get_numeric_data().columns)
categorical_data = list(set(cols) - set(num_data) - set(['credit_risk']))
Y = df.credit_risk
X_cat = df[categorical_data]
X_num_data = df[num_data]

factorize_cols = ['status', 'savings', 'employment_duration', 'installment_rate', 'present_residence', 'number_credi

dummies_cols = list(set(categorical_data) - set(factorize_cols))
```

```
cat_status = ["no checking account", "... < 0 DM", "0<= ... < 200 DM", "... >= 200 DM / salary for at least 1 year"]
cat_savings = ["unknown/no savings account", "... <  100 DM", "100 <= ... <  500 DM", "500 <= ... < 1000 DM", "... >
cat_employment_duration = ["unemployed", "< 1 yr", "1 <= ... < 4 yrs", "4 <= ... < 7 yrs", ">= 7 yrs"]
cat_installment_rate = [">= 35", "25 <= ... < 35", "20 <= ... < 25", "< 20"]
cat_present_residence = ["< 1 yr", "1 <= ... < 4 yrs", "4 <= ... < 7 yrs", ">= 7 yrs"]
cat_number_credits = ["1", "2-3", "4-5", ">= 6"]
cat_people_liable = ["3 or more", "0 to 2"]

X_cat['status'] = pd.factorize(pd.Categorical(X_cat['status'], categories = cat_status))[0]
X_cat['savings'] = pd.factorize(pd.Categorical(X_cat['savings'], categories = cat_savings))[0]
X_cat['employment_duration'] = pd.factorize(pd.Categorical(X_cat['employment_duration'], categories = cat_employment
X_cat['installment_rate'] = pd.factorize(pd.Categorical(X_cat['installment_rate'], categories = cat_installment_rate
X_cat['present_residence'] = pd.factorize(pd.Categorical(X_cat['present_residence'], categories = cat_present_reside
X_cat['number_credits'] = pd.factorize(pd.Categorical(X_cat['number_credits'], categories = cat_number_credits))[0]
X_cat['people_liable'] = pd.factorize(pd.Categorical(X_cat['people_liable'], categories = cat_people_liable))[0]

X_cat= pd.get_dummies(X_cat)

Y = pd.factorize(Y)[0]
```

```
def robust_scaling(df):
    df_robust = df.copy()
    for column in df_robust.columns:
        df_robust[column] = (df_robust[column] - df_robust[column].median())/(df_robust[column].quantile(0.75) - df_
    return df_robust

X_num_data = robust_scaling(X_num_data)


X = pd.concat([X_cat, X_num_data], axis = 1)
```

Importando as bibliotecas de ML

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

2.1 Descrição da amostra de treino e de teste (0,5)

Aplicamos uma taxa de amostragem para teste de 20%, deixando 80% para treino e estratificado para manter a mesma proporçao de bons e mals nas duas amostras.

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=1234, stratify=Y)

log_reg = LogisticRegression()
log = log_reg.fit(X_train, y_train)
y_pred = log.predict(X_train)
y_predtest = log.predict(X_test)
print("train score: ", log.score(X_train,y_train))
print("test score: ", log.score(X_test,y_test))
```

```
train score:  0.7825
test score:  0.775
/Users/karinseeder/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a solver to
silence this warning.
  FutureWarning)
```

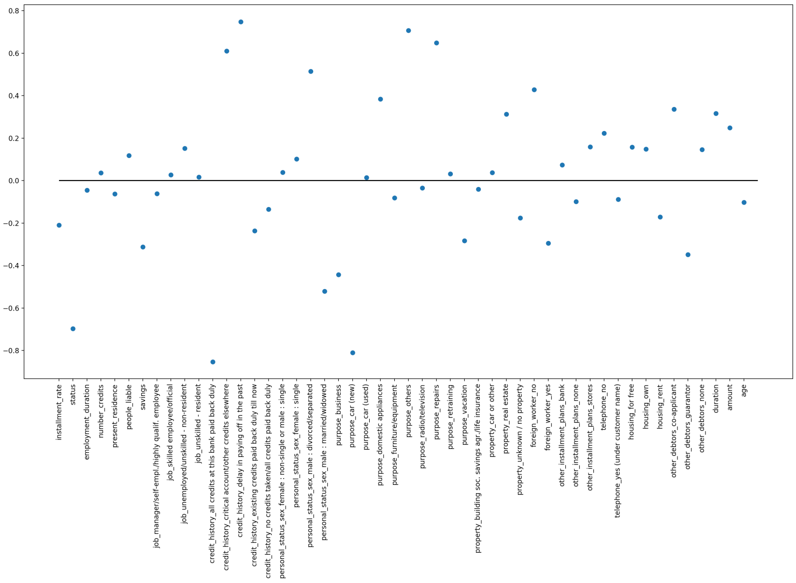
2.2 Descrição do modelo, coeficientes, significância obtida (2,0)

```
odds = np.exp(log.coef_[0])/(1 + np.exp(log.coef_[0]))
coeficientes = pd.DataFrame(odds,
             X_train.columns,
             columns=['coef'])\
            .sort_values(by='coef', ascending=False)
print(coeficientes)
print(" ")
plt.rcParams["figure.figsize"] = (20,10)
plt.plot(log.coef_.T, 'o', label = "teste")
plt.xticks(range(X_train.shape[1]), X_train.columns, rotation=90)
plt.hlines(0,0, X_train.shape[1])
```

```
                                                      coef
credit_history_delay in paying off in the past     0.678705
purpose_others                                     0.669687
purpose_repairs                                    0.656786
credit_history_critical account/other credits e... 0.648019
personal_status_sex_male : divorced/separated      0.625732
foreign_worker_no                                  0.605555
purpose_domestic appliances                        0.594732
other_debtors_co-applicant                         0.583300
duration                                           0.578365
property_real estate                               0.577447
amount                                             0.561871
telephone_no                                       0.555429
other_installment_plans_stores                     0.539598
housing_for free                                   0.539195
job_unemployed/unskilled - non-resident            0.537954
housing_own                                        0.536939
other_debtors_none                                 0.536445
people_liable                                      0.529544
personal_status_sex_female : single                0.525398
other_installment_plans_bank                       0.518406
personal_status_sex_female : non-single or male... 0.509630
property_car or other                              0.509334
number_credits                                     0.509061
purpose_retraining                                 0.508028
job_skilled employee/official                      0.506645
job_unskilled - resident                           0.504119
```

```
purpose_car (used)                                 0.503631
purpose_radio/television                           0.491185
property_building soc. savings agr./life insurance 0.489833
employment_duration                                0.488828
job_manager/self-empl./highly qualif. employee     0.484633
present_residence                                  0.484393
purpose_furniture/equipment                        0.479797
telephone_yes (under customer name)                0.477778
other_installment_plans_none                       0.475346
age                                                0.474306
credit_history_no credits taken/all credits pai... 0.466320
housing_rent                                       0.457244
property_unknown / no property                     0.456291
installment_rate                                   0.447800
credit_history_existing credits paid back duly ... 0.441138
purpose_vacation                                   0.429791
foreign_worker_yes                                 0.426785
savings                                            0.422683
other_debtors_guarantor                            0.413701
purpose_business                                   0.391037
personal_status_sex_male : married/widowed         0.372742
status                                             0.332473
purpose_car (new)                                  0.307781
credit_history_all credits at this bank paid ba... 0.298801
```

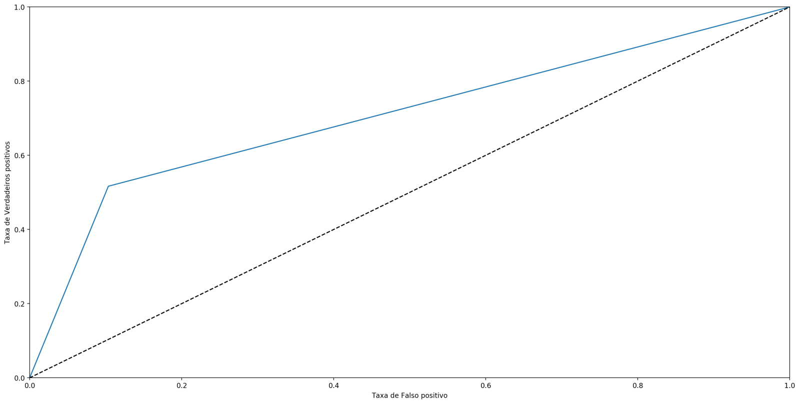
```
<matplotlib.collections.LineCollection at 0x7fa31b6e7ef0>
```




A partir do resultado acima podemos notar que as variáveis mais importantes fazem sentido dado que o histórico de atraso no passado, propósito do empréstimo sem definiçao podendo ser um emprestimo sem garantia, para fins de reparo tambem podem ser creditos mais arriscado dado que não necessariamente a um valor agregado para aquele montante emprestado. Por fim, crédito em outros bancos já mostra um individamento maior do indivíduo, diminuindo sua capacidade de pagamento por já possuir compromisso com outras instituições.

2.3 Curva ROC obtida no treino - análise (1,0)

```python
from sklearn.metrics import roc_curve
plt.rcParams["figure.figsize"] = (20,10)
fpr, tpr, thresholds = roc_curve(y_train, y_pred)

def plot_roc(fpr, tpr, thresholds):
  plt.plot(fpr, tpr, label=None)
  plt.plot([0,1], [0,1], 'k--')
  plt.axis([0,1,0,1])
  plt.xlabel("Taxa de Falso positivo")
  plt.ylabel("Taxa de Verdadeiros positivos")

plot_roc(fpr, tpr, thresholds)
plt.show()
```




2.4 Medidas de desempenho para amostra de treino e para a amostra de teste: Matriz confusão, precisão, revocação, $F1$ (2,5)

```
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
print('treino')
print('matriz de confusão: ')
print(confusion_matrix(y_train, y_pred))
print('Precision: ', precision_score(y_train, y_pred))
print('Recall: ', recall_score(y_train, y_pred))
print('F1: ', f1_score(y_train, y_pred))

print('teste')
print('matriz de confusão: ')
print(confusion_matrix(y_test, y_predtest))
print('Precision: ', precision_score(y_test, y_predtest))
print('Recall: ', recall_score(y_test, y_predtest))
print('F1: ', f1_score(y_test, y_predtest))
```

```
treino
matriz de confusão:
[[502  58]
 [116 124]]
Precision:  0.6813186813186813
Recall:  0.5166666666666667
F1:  0.5876777251184834
teste
matriz de confusão:
[[121  19]
 [ 26  34]]
Precision:  0.6415094339622641
Recall:  0.5666666666666667
F1:  0.6017699115044247
```

# Conclusão

Pelos resultados obtidos podemos concluir que o modelo não possui overffiting. Apesar do ajuste apresentar uma acurária de 78%, a partir dos resultados da matriz de confusão é possível dizer que o modelo não classifica bem os mal pagadores.

---

Published from aula2.pmd using Pweave 0.30.3 on 01-08-2021.