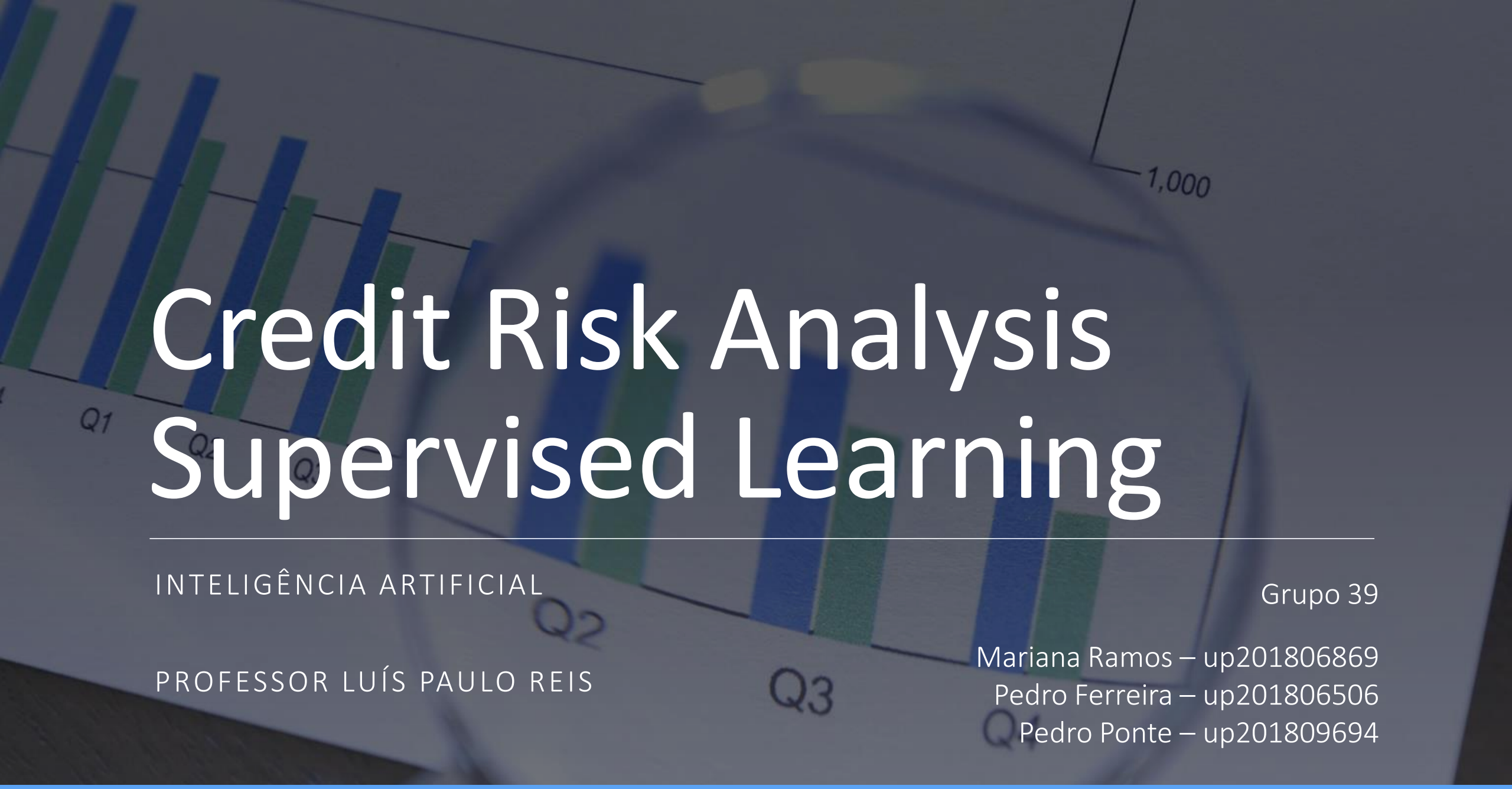



# Credit Risk Analysis Supervised Learning

INTELIGÊNCIA ARTIFICIAL

PROFESSOR LUÍS PAULO REIS

Grupo 39

Mariana Ramos – up201806869
Pedro Ferreira – up201806506
Pedro Ponte – up201809694

# Especificação do Projeto

O objetivo deste trabalho é utilizar *Supervised Learning* para prever se um empréstimo será pago ou não (*default_ind*). Para isso, contamos com um *dataset* de 855969 amostras de empréstimo. Estas amostras contêm os 73 atributos, sendo alguns dos mais importantes:

- **loan_amnt** – Quantidade de dinheiro solicitada pelo cliente;
- **int_rate** – Taxa de juros do empréstimo;
- **grade** – Grau do empréstimo, com valores A, B, C, D, E, F, G. Quanto mais próximo de A for o grau do empréstimo, menor será o risco deste não ser pago;
- **annual_inc** – Rendimento anual do cliente;
- **purpose** – Motivo principal para o pedido de empréstimo;
- **installment** – Valor pago por mês pelo empréstimo;
- **term** – tempo até o empréstimo estar pago.

Tirando partido dos diferentes valores destes atributos, utilizaremos vários classificadores para avaliar o *default_ind* cada empréstimo, com uma taxa de acerto aceitável.

# Detalhes da implementação

**Ferramentas**

Para o desenvolvimento deste projeto iremos usar **Python** e as suas bibliotecas que tornam mais simples o desenvolvimento de projetos de aprendizagem supervisionada.


- **Scikit-learn** – ML Algorithms
- **Keras -** biblioteca dedicada a redes neuronais;
- **Pandas** –análise de dados;
- **Numpy** –análise de dados;
- **Matplolib** - visualização de dados.
- **Seachorn** – visualização de dados

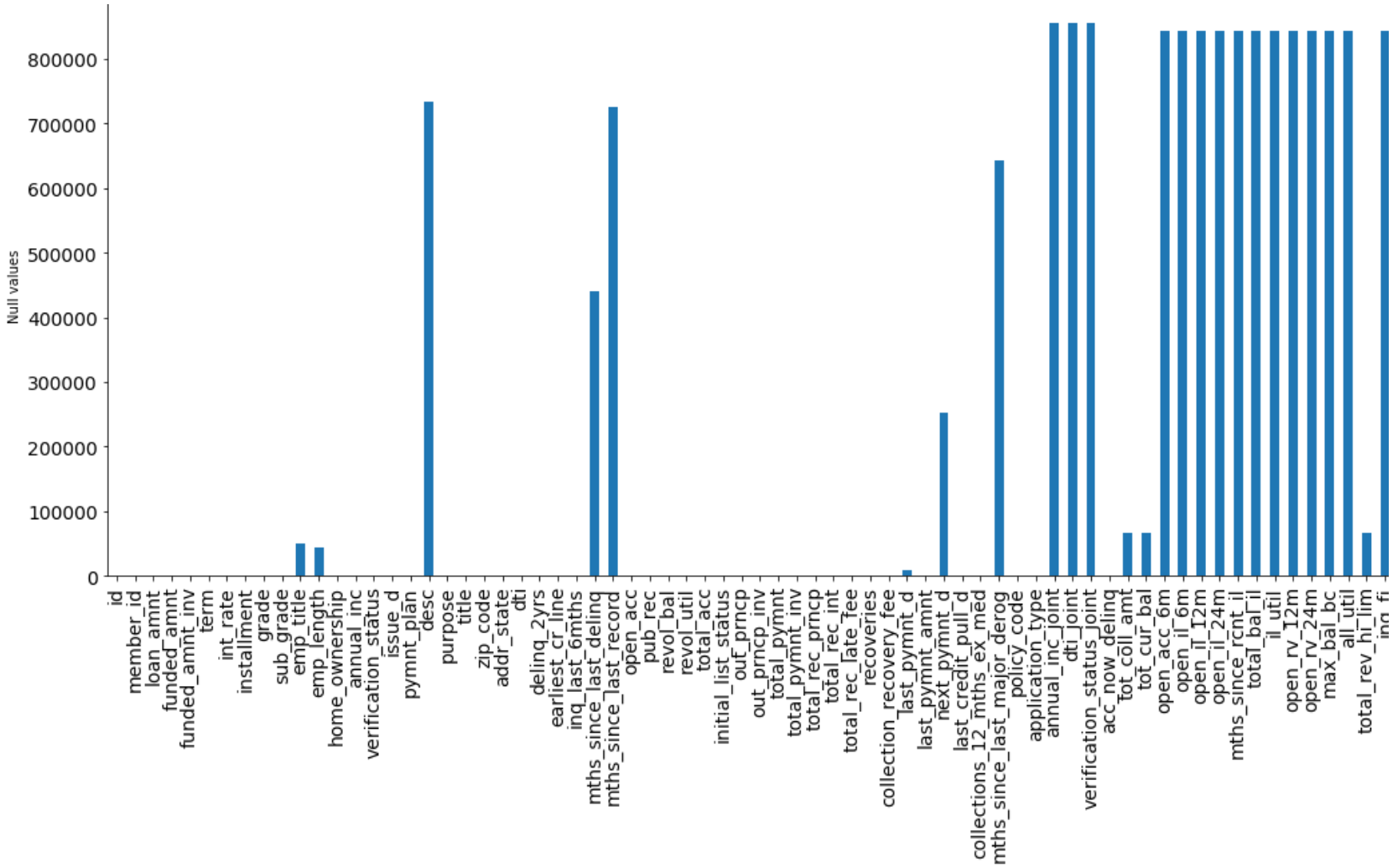
**Ambiente de desenvolvimento -** Jupyter Notebook

**Data Set -** https://www.kaggle.com/rameshmehta/credit-risk-analysis

# Análise de dados

O nosso problema apresenta algumas propriedades:

- Atributos Nominais e Discretos binários
- Dimensão : 73 colunas
- Tamanho: mais de 850.000 registos
- Sem outliers significativos
- Bastantes dados com valores nulos (como mostra a figura ao lado)



# Pré-processamento de dados

Foi feita uma seleção prévia das colunas pertinentes:

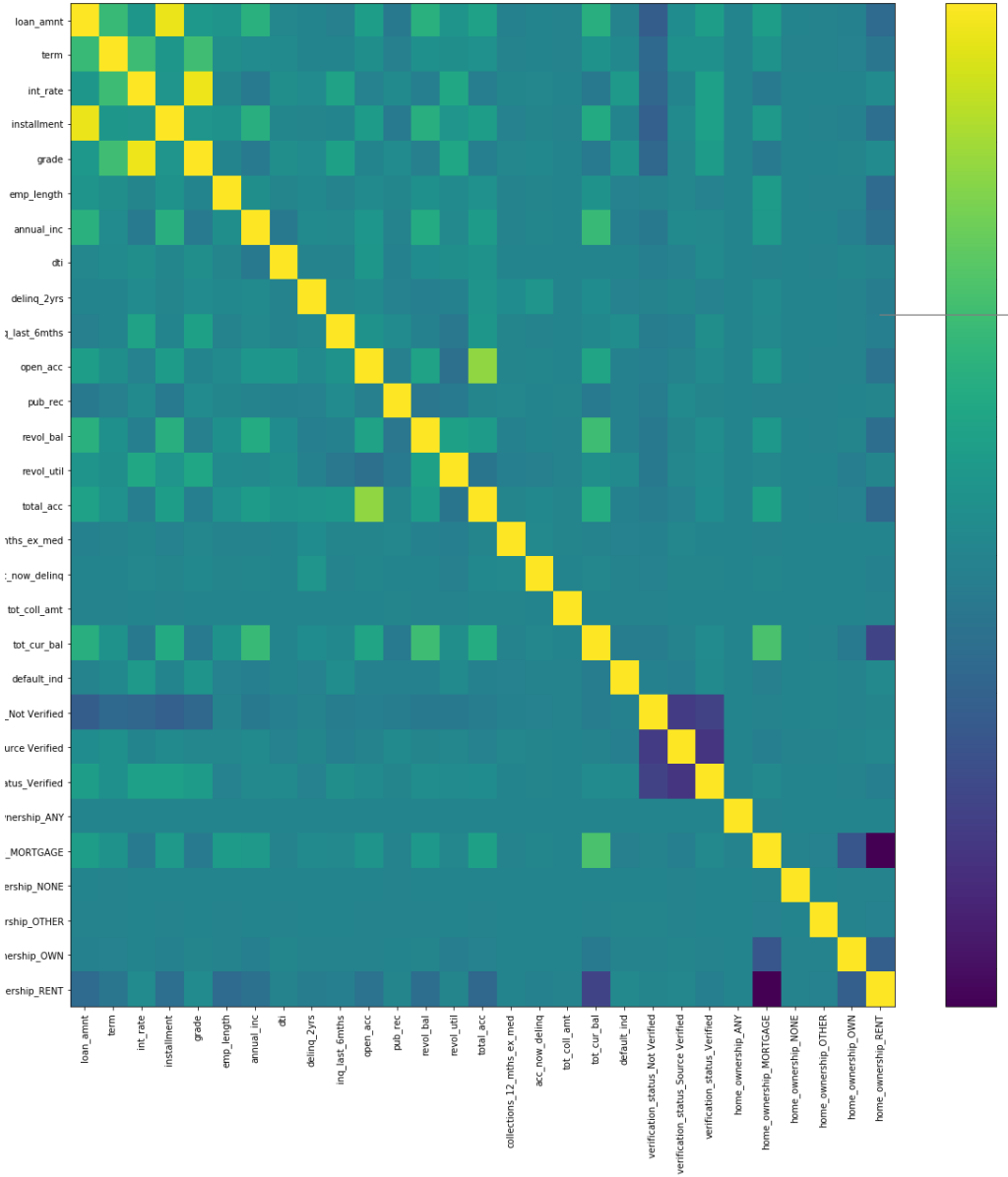
- Removemos as colunas com valores pouco relevantes para a previsão/valores futuros
- Removemos colunas com a maioria dos valores nulos (*verification_status_joint*, *annual_inc_joint*, *dti_joint*, *il_util*, *mths_since_rcnt_il*, *total_bal_il*, *inq_last_12m*, *open_acc_6m* …).
- Colunas com alguns valores a nulo substituímos pela mediana da coluna.

Para além desta limpeza inicial também recorremos à técnica de **encoding**:

Usando a função LabelEncoder da biblioteca ***scikitlearn*** transformamos as colunas *verification_values* e home_*ownership*, mapeando strings para valores numéricos representativos, de modo a poderem ser usados nos algoritmos de supervised learning.

**Divisão dos dados teste e treino**

Depois do pré-processamento dos dados dividimo-los em conjuntos de treino e de teste, com uma percentagem respetiva de 80/20 do número total de linhas.




# Análise de dados

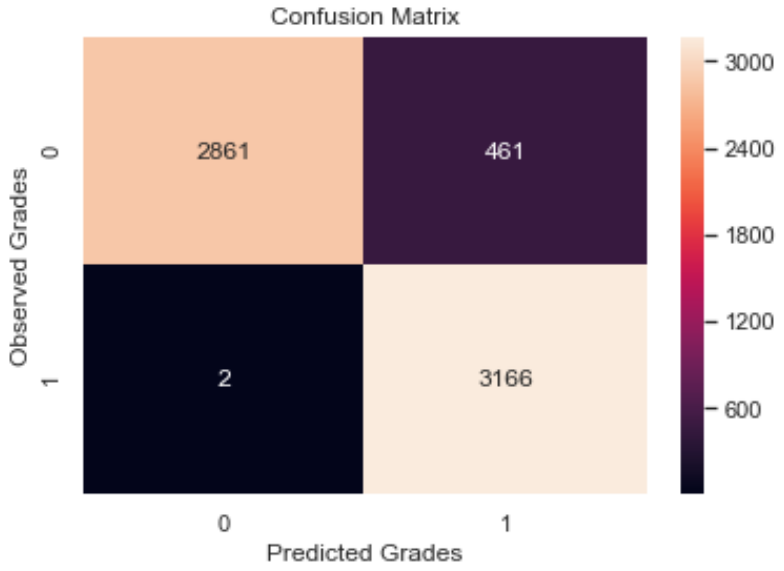
O gráfico ao lado representa a correlação para cada par de atributos do empréstimo, onde a cor mais amarelada corresponde a níveis maiores de correlação e a cor mais roxa a níveis menores.

**NOTA:**

Dado o elevado número de linhas que o *dataset* continha, optamos por utilizar um *subset* com 5% dos dados do *dataset* original. Após selecionar aleatoriamente o *subset*, fazemos ainda um balanceamento dos dados de maneira a que este possua a mesma quantidade de dados com default_*ind* a 0 e a 1, para que não existisse *overfitting*.

Classification report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.86 | 0.93 | 3322 |
| 1 | 0.87 | 1.00 | 0.93 | 3168 |
| accuracy | | | 0.93 | 6490 |
| macro avg | 0.94 | 0.93 | 0.93 | 6490 |
| weighted avg | 0.94 | 0.93 | 0.93 | 6490 |




# Decision Tree

Como técnicas de *tuning* decidimos usar a função *GridSearchCV* com 10 *splits* no nosso processo de treino.

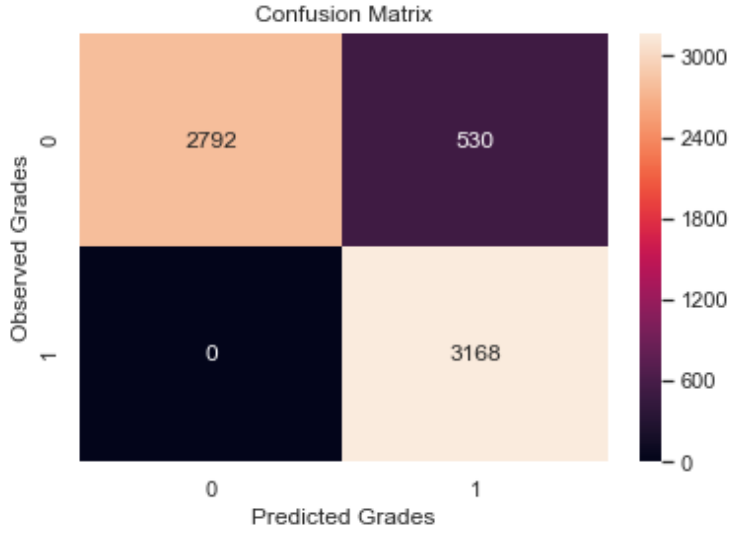
Os melhores parâmetros encontrados foram:

- Criterion: gini
- Max_depth: 19
- Max_features: 10
- Splitter: best;

# K-Nearest Neighbors (k-NN)

Os melhores parâmetros encontrados para este algoritmo foram :

- Algorithm: ball_true
- N_neighbors:5
- Weights:distance



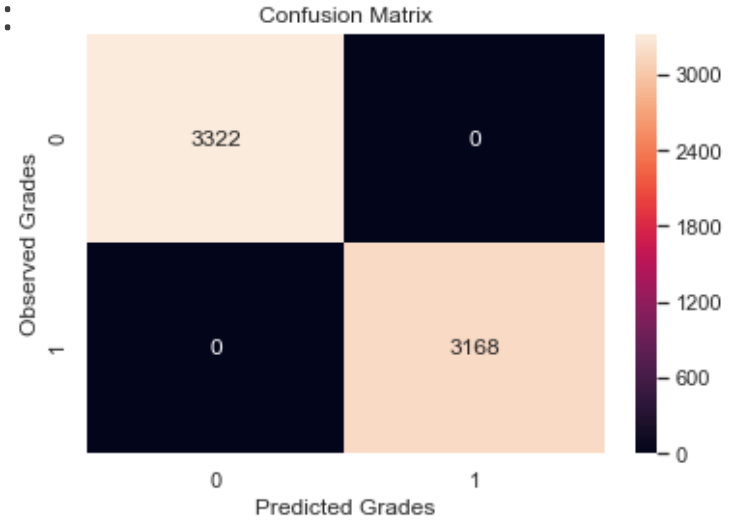

Classification report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.84 | 0.91 | 3322 |
| 1 | 0.86 | 1.00 | 0.92 | 3168 |
| accuracy | | | 0.92 | 6490 |
| macro avg | 0.93 | 0.92 | 0.92 | 6490 |
| weighted avg | 0.93 | 0.92 | 0.92 | 6490 |

# Support Vector Machine (SVM)

Os melhores parâmetros encontrados para este algoritmo foram :

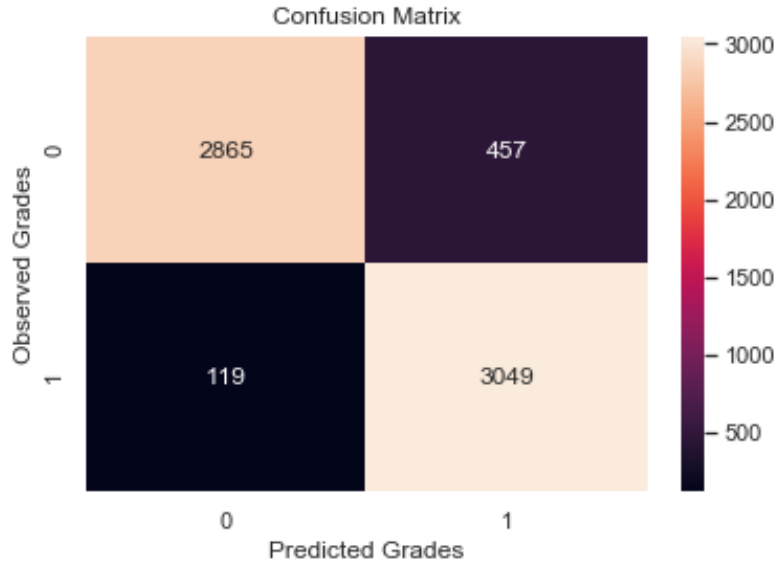
- Kernel: linear




Classification report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 3322 |
| 1 | 1.00 | 1.00 | 1.00 | 3168 |
| accuracy | | | 1.00 | 6490 |
| macro avg | 1.00 | 1.00 | 1.00 | 6490 |
| weighted avg | 1.00 | 1.00 | 1.00 | 6490 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.86 | 0.91 | 3322 |
| 1 | 0.87 | 0.96 | 0.91 | 3168 |
| accuracy | | | 0.91 | 6490 |
| macro avg | 0.91 | 0.91 | 0.91 | 6490 |
| weighted avg | 0.92 | 0.91 | 0.91 | 6490 |




# Neural Networks

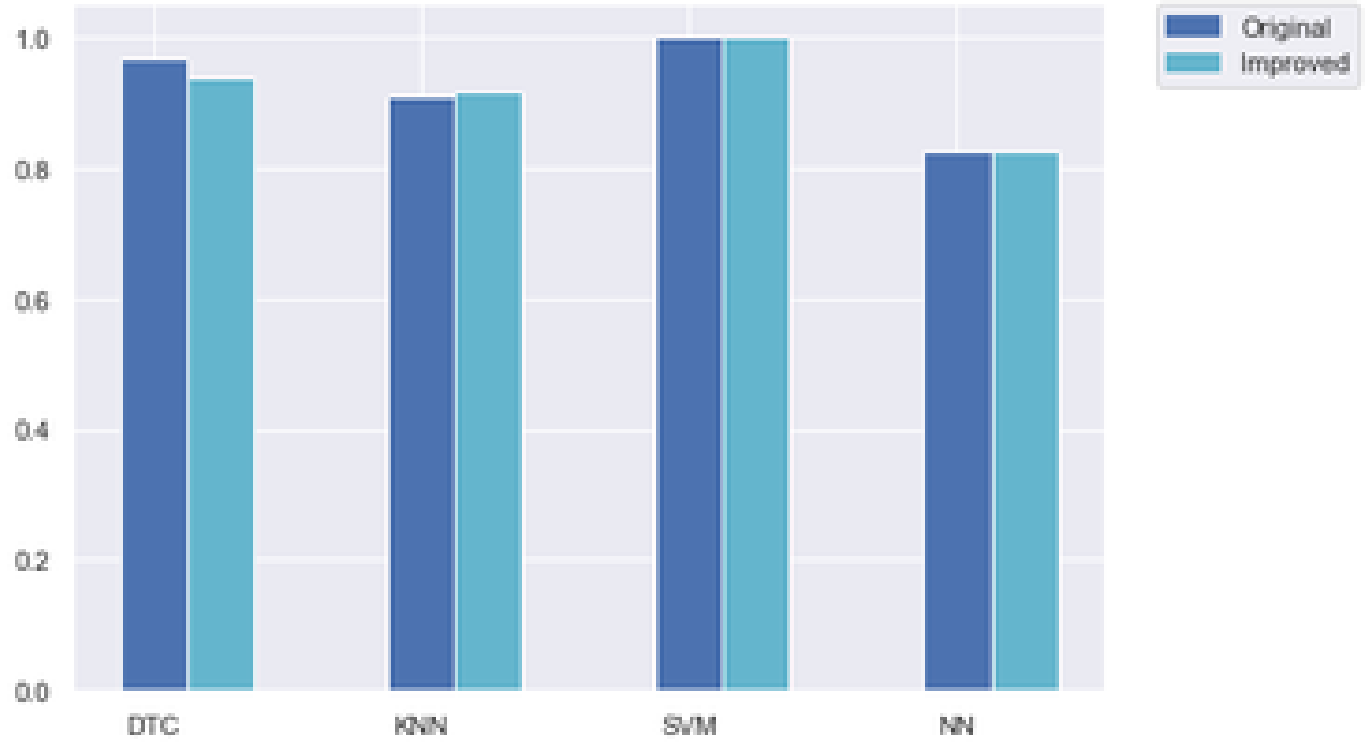
Como técnicas de *tuning* decidimos usar a função *GridSearchCV* com 10 *splits* no nosso processo de treino.

Os melhores parâmetros encontrados foram:

- Activation: tanh
- hidden_layer_size: (100)
- Solver: adam

# Comparação de Algoritmos

Após a implementação de todos os algoritmos concluímos que o Support Vector Machine obtém os melhores resultados apesar de terem sido usados os valores default dos paramêtros e de ser o mais demorado.




| Algorithm | Accuracy Normal | Accuracy Improved |
|---|---|---|
| DTC | 0.9661645422943221 | 0.9386635766705291 |
| KNN | 0.9104673619157976 | 0.917110853611433 |
| SVM | 1 | 1 |
| NN | 0.8237929702587872 | 0.8237929702587872 |

# Referências e Trabalho Relacionado

- https://www.kaggle.com/rameshmehta/credit-risk-analysis
- https://rstudio-pubs-static.s3.amazonaws.com/190551_15f6124632824534b7e397ce7ad2f2b8.html
- https://rstudio-pubs-static.s3.amazonaws.com/263968_5057ec1f5a2e48a89aab7f568fc37ade.html
- Slides das aulas teóricas e fichas realizadas nas aulas teórico-práticas
- https://pandas.pydata.org/, https://scikit-learn.org/stable/, https://numpy.org/, https://matplotlib.org/, https://keras.io/