Salário

Nosso objetivo:

Mostrar quantidade de indivíduos em suas diferenciação dos Salários, conseguir prever de acordo com o perfil da pessoa se ela terá um salário menor ou maior R$3.000,00, usando o atributo salario.

Importação das bibliotecas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import mutual_info_classif,mutual_info_regression,SelectKBest,SelectPercentile
from sklearn.linear_model import LogisticRegression
%matplotlib inline
from sklearn.metrics import classification_report, accuracy_score
```

Obtenção dos Dados

```
df=pd.read_csv('/content/CAGED_GV.csv',sep=';')
df
```

	saldo	idade	horascontratuais	indtrabintermitente	indtrabparcial	salario	indicadoraprendiz	month	year	estado	m
0	1	33	40	0	0	2784	0	1	2019	ES	
1	1	19	44	0	0	1045	0	1	2019	ES	
2	1	16	22	0	0	499	1	1	2019	ES	
3	1	19	44	0	0	1205	0	1	2019	ES	
4	1	19	44	0	0	1205	0	1	2019	ES	
...	...	...	...	...	...	...	...	...	...	...	
317299	1	29	44	0	0	1776	0	11	2019	ES	

317300	1	28	44	0	0	1143	0	11	2019	ES	Y
317301	1	22	44	0	0	1431	0	11	2019	ES	Y
317302	1	39	44	0	0	1143	0	11	2019	ES	Y
317303	1	24	44	0	0	1143	0	11	2019	ES	Y

```
best_df=df
stat_data = best_df.describe(include=[object]).T
stat_data['nulos']= pd.DataFrame(best_df.isnull().sum())
stat_data['percent_nulos']= pd.DataFrame(best_df.isna().mean().round(4) * 100)
stat_data["type"] = pd.Series(best_df.dtypes)
stat_data_objetct=stat_data
stat_data_objetct
```

	count	unique	top	freq	nulos	percent_nulos	type
estado	317304	1	ES	317304	0	0.0	object
municipio	317304	5	Serra	105494	0	0.0	object
desc_subclasse	315075	864	COMÉRCIO VAREJISTA DE MERCADORIAS EM GERAL, CO...	12248	2229	0.7	object
desc_classe	315075	514	RESTAURANTES E OUTROS ESTABELECIMENTOS DE SERV...	16718	2229	0.7	object
desc_grupo	315074	239	RESTAURANTES E OUTROS SERVIÇOS DE ALIMENTAÇÃO ...	16823	2230	0.7	object
desc_divisao	315074	85	COMÉRCIO VAREJISTA	59800	2230	0.7	object
desc_secao	315074	21	COMÉRCIO; REPARAÇÃO DE VEÍCULOS AUTOMOTORES E ...	85691	2230	0.7	object
titulo_ocupacao	317293	1564	Vendedor de com�rcio varejista	21266	11	0.0	object
titulo_familia	317293	517	Operadores do com�rcio em lojas e mercados	38011	11	0.0	object
titulo_subgrupo	317293	177	VENDEDORES E DEMONSTRADORES	38011	11	0.0	object
titulo_subprincipal	317293	44	TRABALHADORES DOS SERVI�OS	61442	11	0.0	object

```
best_df=df
stat_data = best_df.describe(include=[np.number]).T
stat_data['nulos']= pd.DataFrame(best_df.isnull().sum())
stat_data['percent_nulos']= pd.DataFrame(df.isna().mean().round(4) * 100)
stat_data["type"] = pd.Series(df.dtypes)
stat_data["median"] = pd.Series(df.median(numeric_only=True))
stat_data_numbers=stat_data
stat_data_numbers
```

	count	mean	std	min	25%	50%	75%	max	nulos	percent_nulos	type	median
saldo	317304.0	0.044784	0.998998	-1.0	-1.0	1.0	1.0	1.0	0	0.0	int64	1.0
idade	317304.0	32.824780	10.787556	11.0	24.0	31.0	40.0	97.0	0	0.0	int64	31.0
horascontratuais	317304.0	40.963590	7.925689	1.0	44.0	44.0	44.0	44.0	0	0.0	int64	44.0
indtrabintermitente	317304.0	0.011771	0.107854	0.0	0.0	0.0	0.0	1.0	0	0.0	int64	0.0
indtrabparcial	317304.0	0.003949	0.062716	0.0	0.0	0.0	0.0	1.0	0	0.0	int64	0.0

Análise de Nulos

indicadoraprendiz	317304.0	0.032893	0.178356	0.0	0.0	0.0	0.0	1.0	0	0.0	int64	0.0

```
fig, ax = plt.subplots(figsize=(20,5))
sns.heatmap(df.isnull(),
            yticklabels=False,
            cbar=False,
            cmap='viridis',
           ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe008162250>
```

Preencher com base na moda

```
mode_sexo = df['desc_sexo'].mode()[0]
mode_idade = df['idade'].mode()[0]
mode_raca = df['desc_raca'].mode()[0]
mode_instr = df['desc_instr'].mode()[0]
mode_def = df['desc_def'].mode()[0]

print(mode_instr)
print(mode_sexo)
print(mode_idade)
print(mode_raca)
print(mode_def)
```

```
Médio Completo
Homem
24
Parda
Não Deficiente
```

```
#Preenchendo valores nulos com moda (categorica)
df['desc_sexo'] = df['desc_sexo'].fillna(mode_sexo)
df['idade'] = df['idade'].fillna(mode_idade)
df['desc_raca'] = df['desc_raca'].fillna(mode_raca)
df['desc_def'] = df['desc_def'].fillna(mode_raca)
df['desc_instr'] = df['desc_instr'].fillna(mode_instr)
```

Verificando resultado

```
fig, ax = plt.subplots(figsize=(20,5))
sns.heatmap(df.isnull(),
            yticklabels=False,
            cbar=False,
            cmap='viridis',
           ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdffd68bd90>
```

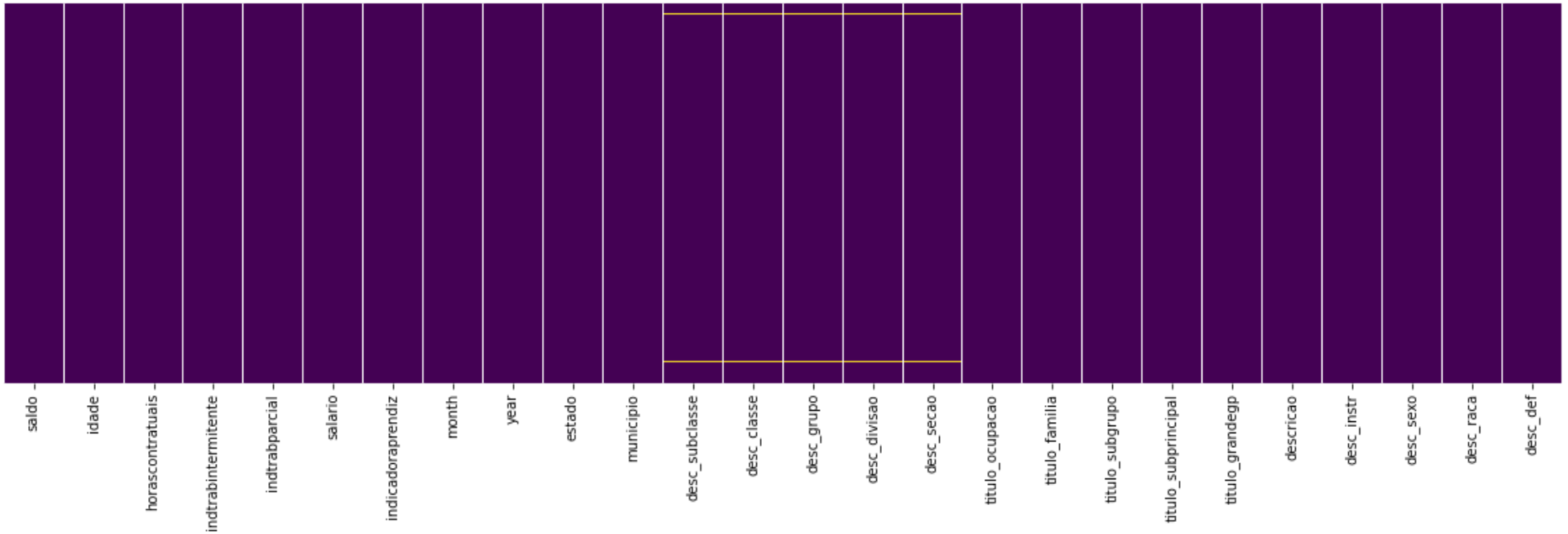
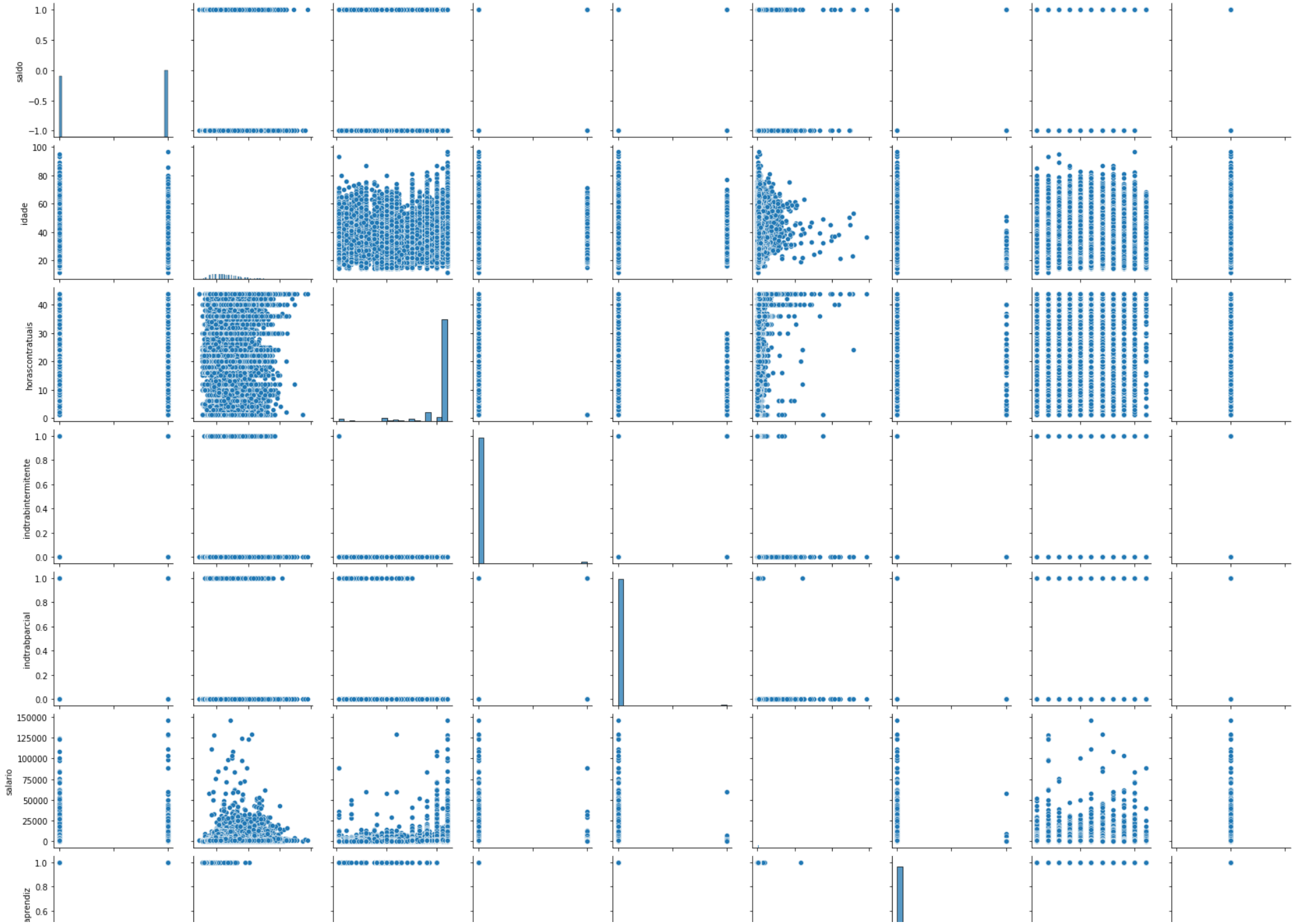
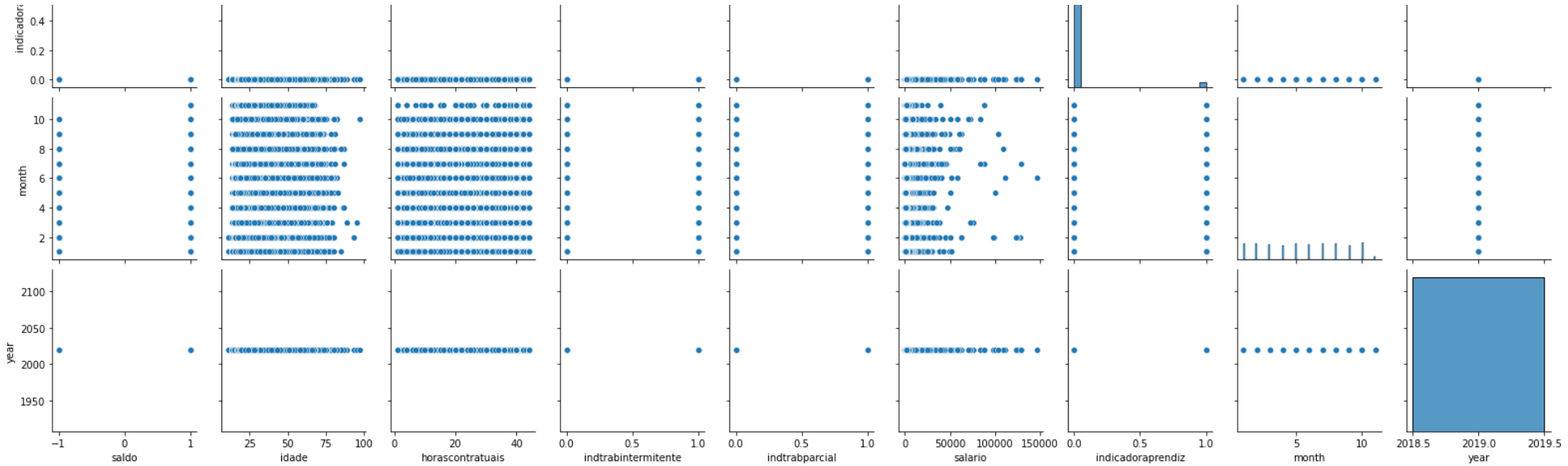


Grafico de visão geral

```
sns.pairplot(df)
```

<seaborn.axisgrid.PairGrid at 0x7fdffd6a6890>





Transformação de dados

```
import sys
df['class_salario']=pd.cut(df['salario'], bins=[0, 3000, sys.maxsize], labels=[0, 1])
```

Categorizando numericamente os dados

```
from sklearn.preprocessing import LabelEncoder
sexo_make = LabelEncoder()
df["desc_sexo_code"] = sexo_make.fit_transform(df["desc_sexo"])
raca_make = LabelEncoder()
df["desc_raca_code"] = raca_make.fit_transform(df["desc_raca"])
instr_make = LabelEncoder()
df["desc_instr_code"] = instr_make.fit_transform(df["desc_instr"])
def_make = LabelEncoder()
df["def_code"] = def_make.fit_transform(df["desc_def"])
df.columns
print(sexo_make.classes_)
print(raca_make.classes_)
print(def_make.classes_)
print(instr_make.classes_)
```

```
['Homem' 'Mulher']
['Amarela' 'Branca' 'Indígena' 'Não Identificado' 'Parda' 'Preta']
['Auditiva' 'Física' 'Intelectual (Mental)' 'Múltipla' 'Não Deficiente'
 'Parda' 'Reabilitado' 'Visual']
['5ª Completo Fundamental' '6ª a 9ª Fundamental' 'Analfabeto'
 'Até 5ª Incompleto' 'Fundamental Completo' 'Médio Completo'
 'Médio Incompleto' 'Superior Completo' 'Superior Incompleto']
```

Categorizando o Target

```
df_label_encoder = df.copy()
target_make = LabelEncoder()
df.insert (6, "class_salario_target", target_make.fit_transform(df["class_salario"]))
```

```
df
```

	saldo	idade	horascontratuais	indtrabintermitente	indtrabparcial	salario	class_salario_target	indicadoraprendiz
0	1	33	40	0	0	2784	0	0
1	1	19	44	0	0	1045	0	0
2	1	16	22	0	0	499	0	1
3	1	19	44	0	0	1205	0	0
4	1	19	44	0	0	1205	0	0
...	...	...	...	...	...	...	...	...
317299	1	29	44	0	0	1776	0	0

317300	1	28	44	0	0	1143	0	0
317301	1	22	44	0	0	1431	0	0
317302	1	39	44	0	0	1143	0	0
317303	1	24	44	0	0	1143	0	0

317304 rows × 32 columns

Balanceando dataset

```
target = 'class_salario'
count_class_0, count_class_1 = df[target].value_counts()
df[target].value_counts()
```

```
0    295749
1     17054
Name: class_salario, dtype: int64
```

```
df_class_0 = df[df[target] == 0]
df_class_1 = df[df[target] == 1]
```

Over-sampling

Unindo registros sampler (classe1) e registros da classe 5

```
df_class_1_over = df_class_1.sample(count_class_0, replace=True,random_state=2)
df_class_1_over
```

	saldo	idade	horascontratuais	indtrabintermitente	indtrabparcial	salario	class_salario_target	indicadoraprendiz
51638	-1	49	44	0	0	9801	1	0
127617	1	37	44	0	0	15188	1	0
221662	1	48	44	0	0	4206	1	0
21100	-1	27	39	0	0	3327	1	0
275355	-1	33	35	0	0	3410	1	0

```
df_test_over = pd.concat([df_class_0, df_class_1_over], axis=0)
df_test_over
```

	saldo	idade	horascontratuais	indtrabintermitente	indtrabparcial	salario	class_salario_target	indicadoraprendiz
0	1	33	40	0	0	2784	0	0
1	1	19	44	0	0	1045	0	0
2	1	16	22	0	0	499	0	1
3	1	19	44	0	0	1205	0	0
4	1	19	44	0	0	1205	0	0
...	...	...	...	...	...	...	...	...
31374	-1	33	24	0	0	3596	1	0
208829	-1	34	40	0	0	11414	1	0

112005	-1	49	44	0	0	3062	1	0
117476	-1	45	36	0	0	28349	1	0

```
df_test_over.columns
```

```
Index(['saldo', 'idade', 'horascontratuais', 'indtrabintermitente',
       'indtrabparcial', 'salario', 'class_salario_target',
       'indicadoraprendiz', 'month', 'year', 'estado', 'municipio',
       'desc_subclasse', 'desc_classe', 'desc_grupo', 'desc_divisao',
       'desc_secao', 'titulo_ocupacao', 'titulo_familia', 'titulo_subgrupo',
       'titulo_subprincipal', 'titulo_grandegp', 'descricao', 'desc_instr',
       'desc_sexo', 'desc_raca', 'desc_def', 'class_salario', 'desc_sexo_code',
       'desc_raca_code', 'desc_instr_code', 'def_code'],
      dtype='object')
```

Seleção de Características

```
initial_selection = df_test_over[['idade', 'desc_sexo_code', 'desc_raca_code','def_code', 'class_salario', 'class_salario_target']]
```

```
initial_selection.columns[:-2]
```

```
Index(['idade', 'desc_sexo_code', 'desc_raca_code', 'def_code'], dtype='object')
```

```
df = initial_selection
target_number = 'class_salario_target'
print("""NAO SERAO UTILIZADOS:
""",df.columns[:-2])
```

```
lista_drop_atributes = df.columns[-2:]
X=df.drop(lista_drop_atributes,axis=1)
y=df[target]

print("""

==> SERAO UTILIZADOS \n ==> Features %s \n ==> target %s """ %(list(df.columns[:-2]),target))
lista_drop_atributes = df.columns[:-1]
```

```
     NAO SERAO UTILIZADOS:
      Index(['idade', 'desc_sexo_code', 'desc_raca_code', 'def_code'], dtype='object')


     ==> SERAO UTILIZADOS
      ==> Features ['idade', 'desc_sexo_code', 'desc_raca_code', 'def_code']
      ==> target class_salario
```

```
from sklearn.feature_selection import chi2
sel_ = SelectKBest(chi2,k=4).fit(X,y)
best_attibutes = list(X.columns[sel_.get_support()])
best_df= df[best_attibutes]
print(best_attibutes)
```

```
     ['idade', 'desc_sexo_code', 'desc_raca_code', 'def_code']
```

```
chi2_s = pd.Series(sel_.scores_)
chi2_s.index = X.columns
chi2_s = chi2_s.sort_values(ascending=False)


chi_columns_priority_order = list(chi2_s.index)+['class_salario']+['class_salario_target']
chi2_s
```

```
     idade             170219.985480
     desc_raca_code     11118.078533
     desc_sexo_code      1621.552238
```

```
def_code               3.250053
dtype: float64
```

```
df_chi2_s = df[chi_columns_priority_order]
df_chi2_s
```

	idade	desc_raca_code	desc_sexo_code	def_code	class_salario	class_salario_target
0	33	1	1	4	0	0
1	19	4	0	4	0	0
2	16	4	1	4	0	0
3	19	4	0	4	0	0
4	19	4	0	4	0	0
...	...	...	...	...	...	...
31374	33	4	0	4	1	1
208829	34	4	0	4	1	1
112005	49	4	0	4	1	1
117476	45	4	0	4	1	1
47259	52	4	1	4	1	1

591498 rows × 6 columns

```
chi2 = chi2_s.sort_values(ascending=False).plot.bar(figsize=(11,5))
```

Resume dataset best features chi2_score

```
stat_data = df_chi2_s.describe(include='all').T
stat_data['nulos']= pd.DataFrame(best_df.isnull().sum())
stat_data['percent_nulos']= pd.DataFrame(df.isna().mean().round(4) * 100)
stat_data["type"] = pd.Series(df.dtypes)
stat_data["median"] = pd.Series(df.median(numeric_only=True))
stat_data["skewness"] = pd.Series(df.skew(numeric_only=True))
stat_data["kurtosis"] = pd.Series(df.kurt(numeric_only=True))
stat_data
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max	nulos	percent nulos	t
class_salario	591498.0	2.0	1.0	295749.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	cate	

Criando Train e Test groups do dataset [Features e Target] para aplicar

```
df_chi2_s.columns
df= df_chi2_s
```

```
X_train_columns = df.columns[0:-2]
target = 'class_salario'


X_train,X_test,y_train,y_test = train_test_split(
    df[X_train_columns],
    df[target],
    random_state=0)
print("Dados treino {}".format(X_train.shape))
print("Dados teste {}".format(X_test.shape))
```

```
Dados treino (443623, 4)
Dados teste (147875, 4)
```

```
X_train[X_train_columns]
```

	idade	desc_raca_code	desc_sexo_code	def_code
56613	56	4	1	4
46475	36	4	1	4
87817	32	1	1	4
20751	37	4	0	4
241507	25	4	0	4

Regressão Logística

```
XT = X_train[X_train_columns][:]
XTt = X_test[X_train_columns][:]
clf = LogisticRegression(random_state=0, max_iter=1000,C=3).fit(XT, y_train)
normal_train_resultado = clf.score(XT,y_train)
print("Score de Acuracia para todas caracteristicas [Treino]:",normal_train_resultado)
normal_test_resultado = clf.score(XTt,y_test)
print("Score de Acuracia para todas caracteristicas  [Teste]:",normal_test_resultado)
```

```
Score de Acuracia para todas caracteristicas [Treino]: 0.6382829564743037
Score de Acuracia para todas caracteristicas  [Teste]: 0.6387962806424344
```

Probabilidade para cada classe

```
import seaborn as sns
from sklearn import metrics
r_predict_proba = clf.predict_proba(X_test.iloc[:, :])
r_predict = clf.predict(X_test.iloc[:, :])
r_y_test = list(y_test)
r_inter = clf.intercept_
```

```
#Probabilidade (Predict_proba)
for i in range(len(r_predict)):
    print("""
    ValorReal=%s, Predicted=%s ,
    Predict_proba (no) =%s , Predict_proba (yes) =%s """ %(r_y_test[i], r_predict[i],round(r_predict_proba[i][0], 4),round(r_predict_
```

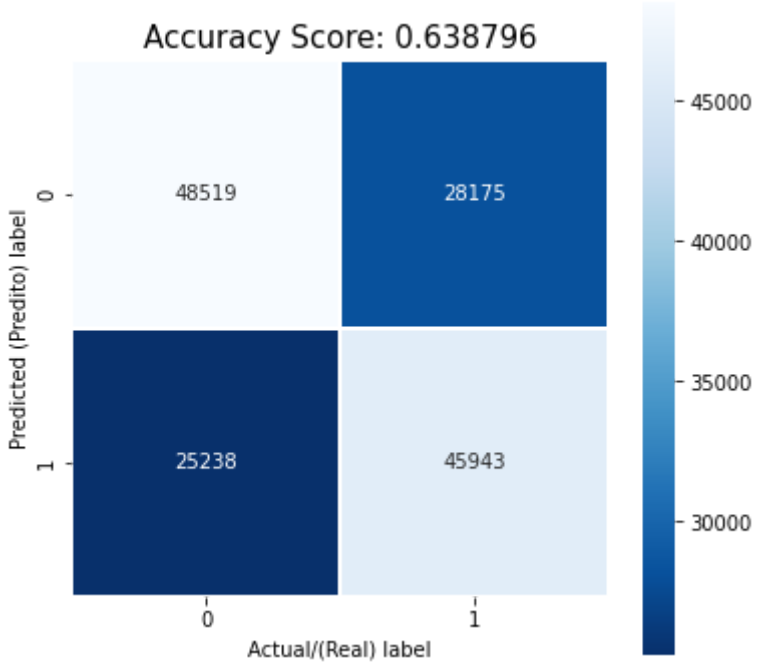
A saída de streaming foi truncada nas últimas 5000 linhas.

```
    ValorReal=0, Predicted=0 ,
    Predict_proba (no) =0.5778 , Predict_proba (yes) =0.4222

    ValorReal=1, Predicted=0 ,
    Predict_proba (no) =0.7246 , Predict_proba (yes) =0.2754

    ValorReal=0, Predicted=0 ,
    Predict_proba (no) =0.7246 , Predict_proba (yes) =0.2754

    ValorReal=0, Predicted=0 ,
    Predict_proba (no) =0.5144 , Predict_proba (yes) =0.4856

    ValorReal=0, Predicted=0 ,
    Predict_proba (no) =0.7686 , Predict_proba (yes) =0.2314

    ValorReal=1, Predicted=1 ,
    Predict_proba (no) =0.4615 , Predict_proba (yes) =0.5385

    ValorReal=1, Predicted=1 ,
    Predict_proba (no) =0.2122 , Predict_proba (yes) =0.7878

    ValorReal=1, Predicted=0 ,
    Predict_proba (no) =0.5291 , Predict_proba (yes) =0.4709

    ValorReal=0, Predicted=0 ,
    Predict_proba (no) =0.5544 , Predict_proba (yes) =0.4456

    ValorReal=1, Predicted=0 ,
    Predict_proba (no) =0.5635 , Predict_proba (yes) =0.4365

    ValorReal=1, Predicted=1 ,
    Predict_proba (no) =0.2818 , Predict_proba (yes) =0.7182
```

```
ValorReal=0, Predicted=0 ,
Predict_proba (no) =0.5146 , Predict_proba (yes) =0.4854

ValorReal=1, Predicted=1 ,
Predict_proba (no) =0.4417 , Predict_proba (yes) =0.5583

ValorReal=0, Predicted=1 ,
Predict_proba (no) =0.4644 , Predict_proba (yes) =0.5356

ValorReal=1, Predicted=1 ,
Predict_proba (no) =0.306 , Predict_proba (yes) =0.694

ValorReal=1, Predicted=0 ,
Predict_proba (no) =0.5945 , Predict_proba (yes) =0.4055

ValorReal=0, Predicted=0 ,
Predict_proba (no) =0.6755 , Predict_proba (yes) =0.3245

ValorReal=1, Predicted=0 ,
Predict_proba (no) =0.7006 , Predict_proba (yes) =0.2994

ValorReal=0, Predicted=0 ,
Predict_proba (no) =0.5146 , Predict_proba (yes) =0.4854

ValorReal=1, Predicted=1 ,
```

Matriz de Confusão

```
CM = metrics.confusion_matrix(r_y_test, r_predict)
plt.figure(figsize=(6,6))
sns.heatmap(CM.T, annot=True, fmt=".0f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.xlabel('Actual/(Real) label');
plt.ylabel('Predicted (Predito) label');
all_sample_title = 'Accuracy Score: %.6f' % (normal_test_resultado)
plt.title(all_sample_title, size = 15);
plt.savefig('logistic_results.png')
```




```
# Cross Validation Classification LogLoss
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
kfold = model_selection.KFold(n_splits=2, random_state=0, shuffle=True)
model = clf
scoring = 'accuracy'
results = model_selection.cross_val_score(model, XT, y_train, cv=kfold, scoring=scoring)
print("accuracy: %.3f (%.3f)" % (results.mean(), results.std()))
test_size = 0.33
model.fit(XT, y_train)
predicted_d = model.predict(XTt)
matrix = confusion_matrix(y_test, predicted_d)
report = classification_report(y_test, predicted_d)
print("=============== Report ================== ")
print(report)
```