

Instalação da researchpy

In [134]:

```
!pip install researchpy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: researchpy in /usr/local/lib/python3.7/dist-packages (0.3.5)
Requirement already satisfied: patsy in /usr/local/lib/python3.7/dist-packages (from researchpy) (0.5.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from researchpy) (1.3.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from researchpy) (1.21.6)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (from researchpy) (0.12.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from researchpy) (1.7.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->researchpy) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->researchpy) (2022.2.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->researchpy) (1.15.0)
```

Importação das bibliotecas

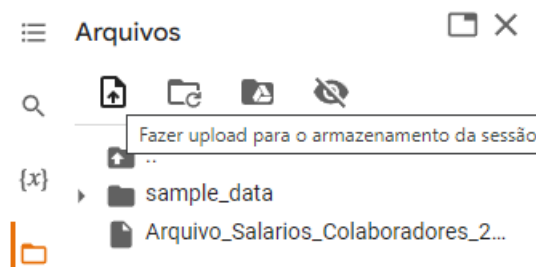
In [135]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import statsmodels.api as sm
import researchpy
import pydotplus
from scipy import stats
from scipy.stats import chi2_contingency, chi2
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score, classification_report
```


Obtendo a base

Como foi utilizado o Google Colab você precisa importar a base: Arquivo_Salarios_Colaboradores_2021.csv

Para realizar isso segue um tutorial:



Selecionar o arquivo:

 Arquivo_Salarios_Colaboradores_2021	05/09/2022 19:44	Arquivo de Valore...	7 KB
---	------------------	----------------------	------

Feito isso podemos então ler os dados a seguir:

Obtendo os dados e lendo o arquivo:

In [136]:

```
salarios = pd.read_csv(r'Arquivo_Salarios_Colaboradores_2021.csv', sep=',')
```

Verificando como os dados estão:

verificando como os dados estão:

In [137]:

```
salarios.head()
```

Out[137]:

	Ordem	salario	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavalia
0	1	8000.80	25	4	11	1	1	1	0	79.38
1	2	8500.17	24	5	11	0	0	1	0	84.13
2	3	3350.59	22	1	12	0	0	0	0	46.15
3	4	9500.24	28	4	14	1	0	0	1	83.85
4	5	1500.63	18	2	12	0	0	0	1	73.64

Interpretação dos dados

Essa base indica que foi dada uma nota de avaliação para funcionários que possuem os dados de:

- Ordem que aparenta ser uma avaliação do funcionário para ordenar ele na empresa
- Sálario
- Idade do funcionário
- O tempo que ele trabalha na empresa
- O total de tempo de estudo
- Se o funcionário participou de projetos estratégicos
- Se o funcionário participou de projetos sustentáveis
- Pontuação do funcionários em projetos 6 sigma
- Se o funcionário participou de projetos sociais
- Aparenta que a nota do funcionário está baseada no critério de projetos e tempo que trabalha na empresa.

Visualizando os dados que foram importados

In [138]:

```
salarios.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 10 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Ordem       180 non-null    int64
1   salario     180 non-null    float64
2   idade       180 non-null    int64
3   tempocasa   180 non-null    int64
4   escolar     180 non-null    int64
5   qproj_estra 180 non-null    int64
6   proj_sustent 180 non-null    int64
7   proj_6sigma 180 non-null    int64
8   proj_social 180 non-null    int64
9   notaavalia  180 non-null    float64
dtypes: float64(2), int64(8)
memory usage: 14.2 KB
```

Verificar o total e a proporção de null na base

In [139]:

```
salarios.isnull() # Aqui verificamos o total de null na base se existir
salarios.isnull().sum()/len(salarios) # Aqui obtemos a proporção de null se existir

# Criamos um novo DataFrame para verificar o total e a proporção de null na base
pd.DataFrame(zip(salarios.isnull().sum(),salarios.isnull().sum()/len(salarios)), columns=['Count', 'Proportion'], index=salarios.columns)
```

Out[139]:

	Count	Proportion
Ordem	0	0.0

salario	Count	Proportion
idade	0	0.0
tempocasa	0	0.0
escolar	0	0.0
qproj_estra	0	0.0
proj_sustent	0	0.0
proj_6sigma	0	0.0
proj_social	0	0.0
notaavalia	0	0.0

Aqui realizamos uma análise descritiva dos dados e arredondamos o valor com 2 casas decimais.

Após essa análise conseguimos chegar a algumas percepções dos dados:

- A média de salário da base é de: **8539.49**
- O desvio padrão é de: **4729.51**
- O menor salário da base é de: **1500.63**
- Meu percentil de 25% na base de salários é de: **5491.23** neste caso podemos avaliar que possuímos uma média de salários mencionada anteriormente.
- Meu percentil de 50% na base de salários é de: **7551.12** neste caso podemos avaliar que possuímos uma média de salários mencionada anteriormente.
- Meu percentil de 75% na base de salários é de: **10584.68** neste caso podemos avaliar que possuímos uma média de salários mencionada anteriormente.
- O maior salário da base é de: **25329.91**

Ao que tudo indica ter mais idade demonstra que você irá receber um salário maior. Em consequência da idade você tem mais tempo de empresa e também sua nota de avaliação é maior que as dos demais.

In [140]:

```
# descritivo das variáveis: medidas resumo
salarios.describe().round(2).T
```

Out[140]:

	count	mean	std	min	25%	50%	75%	max
Ordem	180.0	90.50	52.11	1.00	45.75	90.50	135.25	180.00
salario	180.0	8539.49	4729.51	1500.63	5491.23	7551.12	10584.68	25329.91
idade	180.0	31.07	9.35	18.00	24.00	28.00	36.00	65.00
tempocasa	180.0	9.07	5.61	1.00	5.00	7.00	13.00	25.00
escolar	180.0	12.58	2.57	7.00	11.00	13.00	14.00	19.00
qproj_estra	180.0	1.27	1.22	0.00	1.00	1.00	1.00	7.00
proj_sustent	180.0	0.28	0.45	0.00	0.00	0.00	1.00	1.00
proj_6sigma	180.0	0.55	0.50	0.00	0.00	1.00	1.00	1.00
proj_social	180.0	0.50	0.50	0.00	0.00	0.50	1.00	1.00
notaavalia	180.0	71.74	15.99	34.48	60.41	72.08	83.85	98.96

Histograma da variável Salário

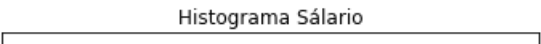
Aqui vemos a frequência do salário e ao analisar o histograma percebemos que existe realmente uma frequência de salários entre **5000 à 10.000**.

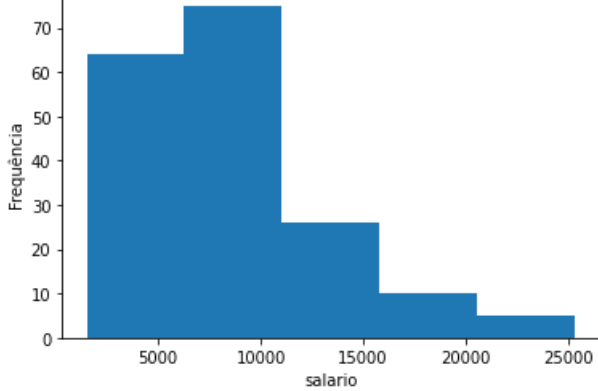
In [141]:

```
plt.hist(salarios['salario'], bins=5)
plt.ylabel('Frequência')
plt.xlabel('salario')
plt.title('Histograma Salário')
```

Out[141]:

Text(0.5, 1.0, 'Histograma Salário')





Verificar as distribuição das variáveis numéricas:

- salario
- idade
- tempocasa
- escolar
- qproj_estra
- proj_sustent
- proj_6sigma
- proj_social
- notaavalia

Fizemos isso para verificar o peso das variáveis.

In [142]:

```
plt.figure(figsize=(24,20))

plt.subplot(4, 2, 1)
fig = salarios['salario'].hist(color='y')
fig.set_xlabel('Valor salário')
fig.set_ylabel('Número de casos')

plt.subplot(4, 2, 2)
fig = salarios['idade'].hist(color='skyblue')
fig.set_xlabel('Idade')
fig.set_ylabel('Número de casos')

plt.subplot(4, 2, 3)
fig = salarios['tempocasa'].hist(color='g')
fig.set_xlabel('Tempo de casa')
fig.set_ylabel('Número de casos')

plt.subplot(4, 2, 4)
fig = salarios['escolar'].hist(color='b')
fig.set_xlabel('Tempo de escolaridade')
fig.set_ylabel('Número de casos')

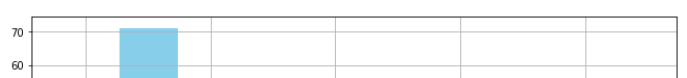
plt.subplot(4, 2, 5)
fig = salarios['qproj_estra'].hist(color='bisque')
fig.set_xlabel('Quantidade de projetos estratégicos')
fig.set_ylabel('Número de casos')

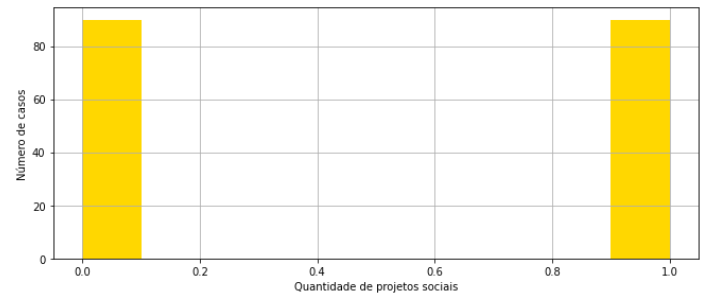
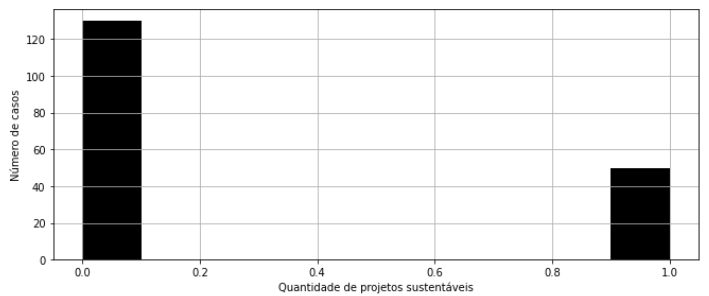
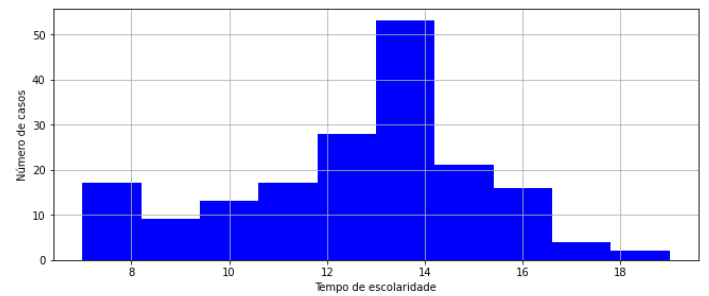
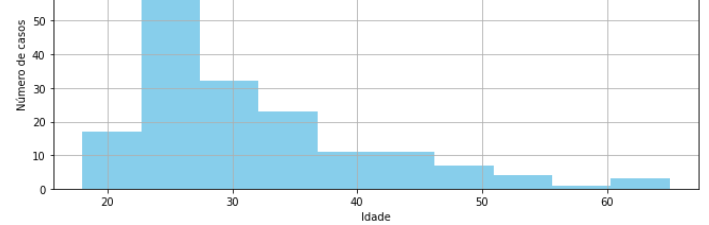
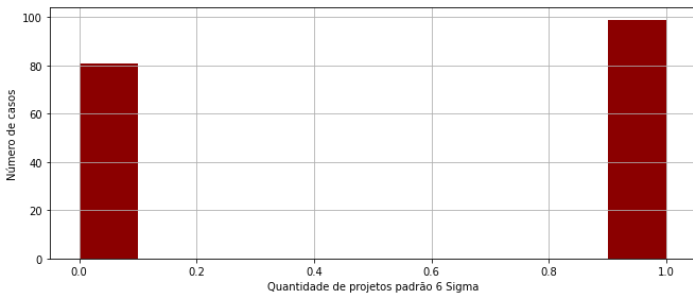
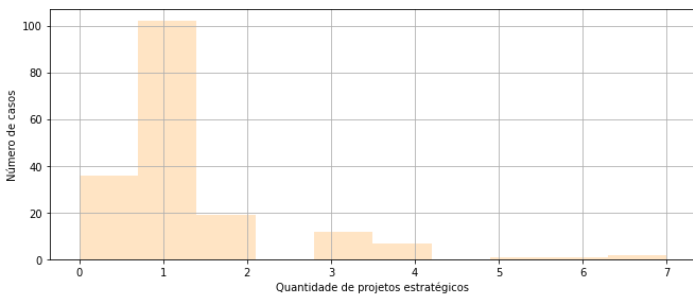
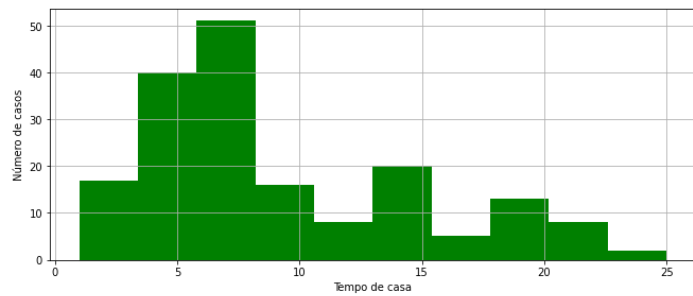
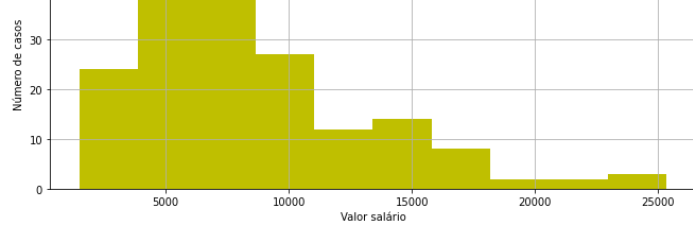
plt.subplot(4, 2, 6)
fig = salarios['proj_sustent'].hist(color='black')
fig.set_xlabel('Quantidade de projetos sustentáveis')
fig.set_ylabel('Número de casos')

plt.subplot(4, 2, 7)
fig = salarios['proj_6sigma'].hist(color='darkred')
fig.set_xlabel('Quantidade de projetos padrão 6 Sigma')
fig.set_ylabel('Número de casos')

plt.subplot(4, 2, 8)
fig = salarios['proj_social'].hist(color='gold')
fig.set_xlabel('Quantidade de projetos sociais')
fig.set_ylabel('Número de casos')

plt.show()
```



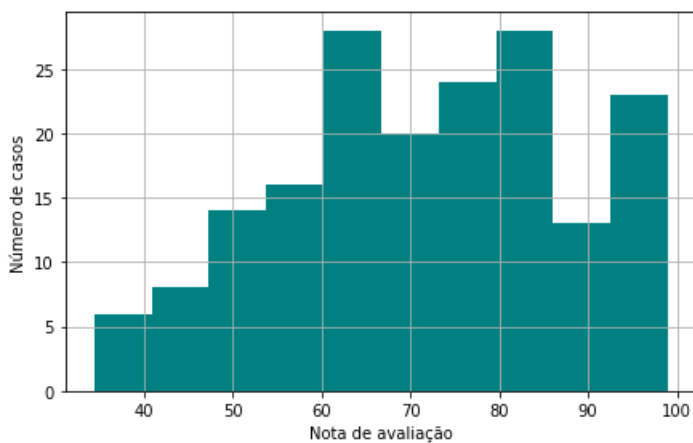


In [143]:

```
plt.figure(figsize=(24,20))

plt.subplot(4,3,11)
fig = salarios['notaavalia'].hist(color='teal')
fig.set_xlabel('Nota de avaliação')
fig.set_ylabel('Número de casos')

plt.show()
```



Criação dos gráficos de Boxplot

Criamos os gráficos a seguir para verificar os Outliers da nossa base com as variáveis:

- salario
- idade
- tempocasa
- projetos

- escolar
- qproj_estra
- proj_sustent
- proj_6sigma
- proj_social
- notaavalua

Baseado no peso das variáveis nos escolhemos verificar as variáveis anteriores.

In [144]:

```
plt.figure(figsize=(24,20))

plt.subplot(4, 2, 1)
fig = salarios.boxplot(column='salario')
fig.set_title('Salário do funcionário')
fig.set_ylabel('Salário')

plt.subplot(4, 2, 2)
fig = salarios.boxplot(column='idade')
fig.set_title('Idade do funcionário')
fig.set_ylabel('Idade')

plt.subplot(4, 2, 3)
fig = salarios.boxplot(column='tempocasa')
fig.set_title('Tempo de casa do funcionário')
fig.set_ylabel('Tempo de casa')

plt.subplot(4, 2, 4)
fig = salarios.boxplot(column='escolar')
fig.set_title('Tempo de estudo do funcionário')
fig.set_ylabel('Tempo de estudo')

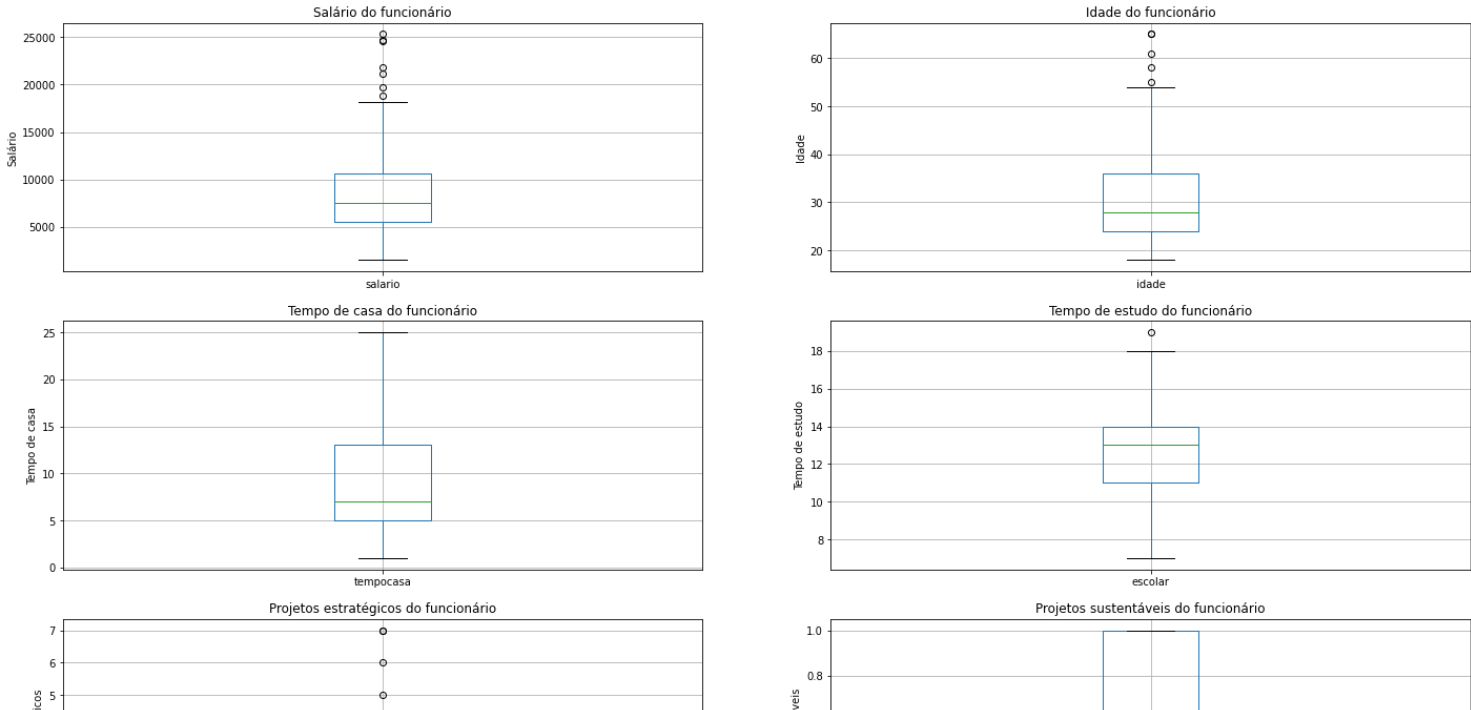
plt.subplot(4, 2, 5)
fig = salarios.boxplot(column='qproj_estra')
fig.set_title('Projetos estratégicos do funcionário')
fig.set_ylabel('Projetos estratégicos')

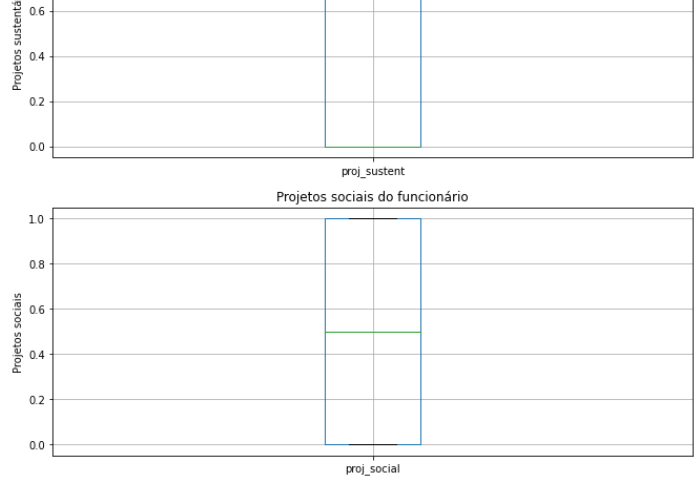
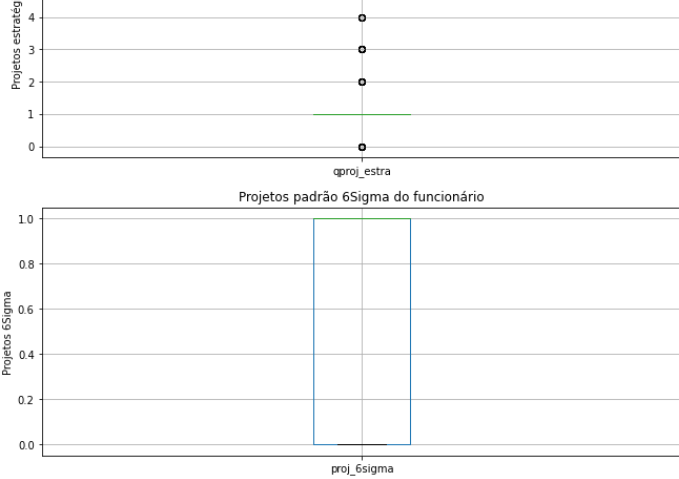
plt.subplot(4, 2, 6)
fig = salarios.boxplot(column='proj_sustent')
fig.set_title('Projetos sustentáveis do funcionário')
fig.set_ylabel('Projetos sustentáveis')

plt.subplot(4, 2, 7)
fig = salarios.boxplot(column='proj_6sigma')
fig.set_title('Projetos padrão 6Sigma do funcionário')
fig.set_ylabel('Projetos 6Sigma')

plt.subplot(4, 2, 8)
fig = salarios.boxplot(column='proj_social')
fig.set_title('Projetos sociais do funcionário')
fig.set_ylabel('Projetos sociais')

plt.show()
```





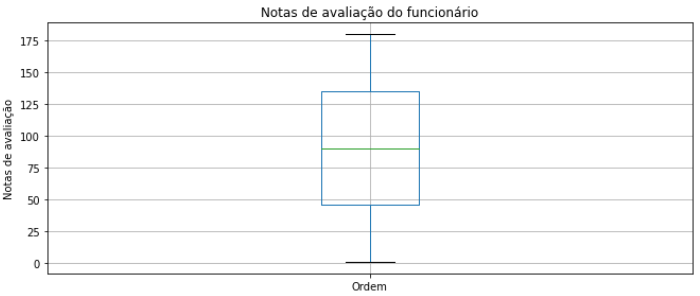
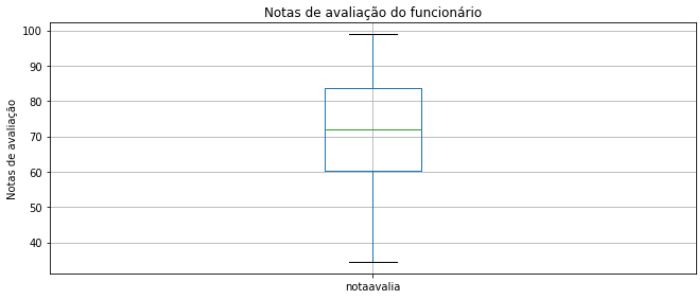
In [145]:

```
plt.figure(figsize=(24,20))

plt.subplot(4, 2, 1)
fig = salarios.boxplot(column='notaavalia')
fig.set_title('Notas de avaliação do funcionário')
fig.set_ylabel('Notas de avaliação')

plt.subplot(4, 2, 2)
fig = salarios.boxplot(column='Ordem')
fig.set_title('Notas de avaliação do funcionário')
fig.set_ylabel('Notas de avaliação')

plt.show()
```



Análise para remoção de coluna

Até este ponto da análise acreditamos que essas colunas não serão necessárias:

- proj_sustent
- proj_6sigma
- proj_social

In [146]:

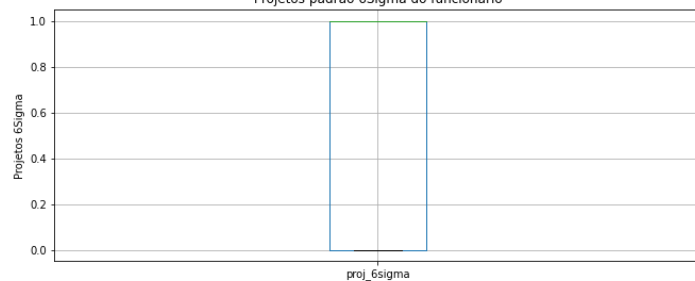
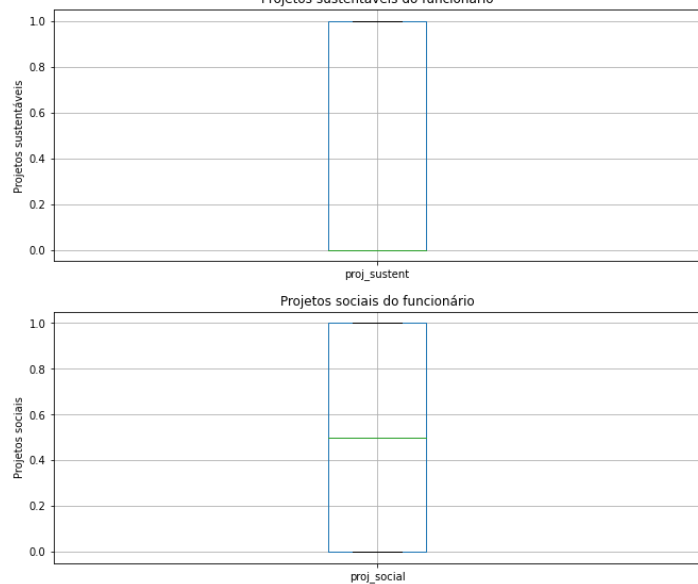
```
plt.figure(figsize=(24,20))

plt.subplot(4, 2, 1)
fig = salarios.boxplot(column='proj_sustent')
fig.set_title('Projetos sustentáveis do funcionário')
fig.set_ylabel('Projetos sustentáveis')

plt.subplot(4, 2, 2)
fig = salarios.boxplot(column='proj_6sigma')
fig.set_title('Projetos padrão 6Sigma do funcionário')
fig.set_ylabel('Projetos 6Sigma')

plt.subplot(4, 2, 3)
fig = salarios.boxplot(column='proj_social')
fig.set_title('Projetos sociais do funcionário')
fig.set_ylabel('Projetos sociais')

plt.show()
```



Remoção da coluna para correlação

A seguir nos removemos a coluna de ordem, pois acreditamos que ela não seja necessária para o cálculo de correlação:

- Ordem

Colocamos em uma nova base para que não perca a original caso precise re-avaliar algo.

In [147]:

```
salariosBase = salarios.drop(['Ordem'], axis=1)
```

In [148]:

```
salariosBase.head(5)
```

Out[148]:

	salario	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavalia
0	8000.80	25	4	11	1	1	1	0	79.38
1	8500.17	24	5	11	0	0	1	0	84.13
2	3350.59	22	1	12	0	0	0	0	46.15
3	9500.24	28	4	14	1	0	0	1	83.85
4	1500.63	18	2	12	0	0	0	1	73.64

In [149]:

```
salarios.head(5)
```

Out[149]:

	Ordem	salario	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavalia
0	1	8000.80	25	4	11	1	1	1	0	79.38
1	2	8500.17	24	5	11	0	0	1	0	84.13
2	3	3350.59	22	1	12	0	0	0	0	46.15
3	4	9500.24	28	4	14	1	0	0	1	83.85
4	5	1500.63	18	2	12	0	0	0	1	73.64

Execução do coeficiente de correlação

Nós iremos utilizar o método de correlação de person, esse coeficiente é um teste que mede a relação estatística entre duas variáveis contínuas. Se a associação entre os elementos não for linear, o coeficiente não será representado adequadamente.

Resultado do coeficiente:

- $\rho = 0,9$ a 1 (positivo ou negativo): correlação muito forte.
- $\rho = 0,7$ a $0,9$ (positivo ou negativo): correlação forte.
- $\rho = 0,5$ a $0,7$ (positivo ou negativo): correlação moderada.
- $\rho = 0,3$ a $0,5$ (positivo ou negativo): correlação fraca.
- $\rho = 0$ a $0,3$ (positivo ou negativo): não possui correlação.

In [150]:

```
corr = salariosBase.corr(method='pearson')
corr
```

Out[150]:

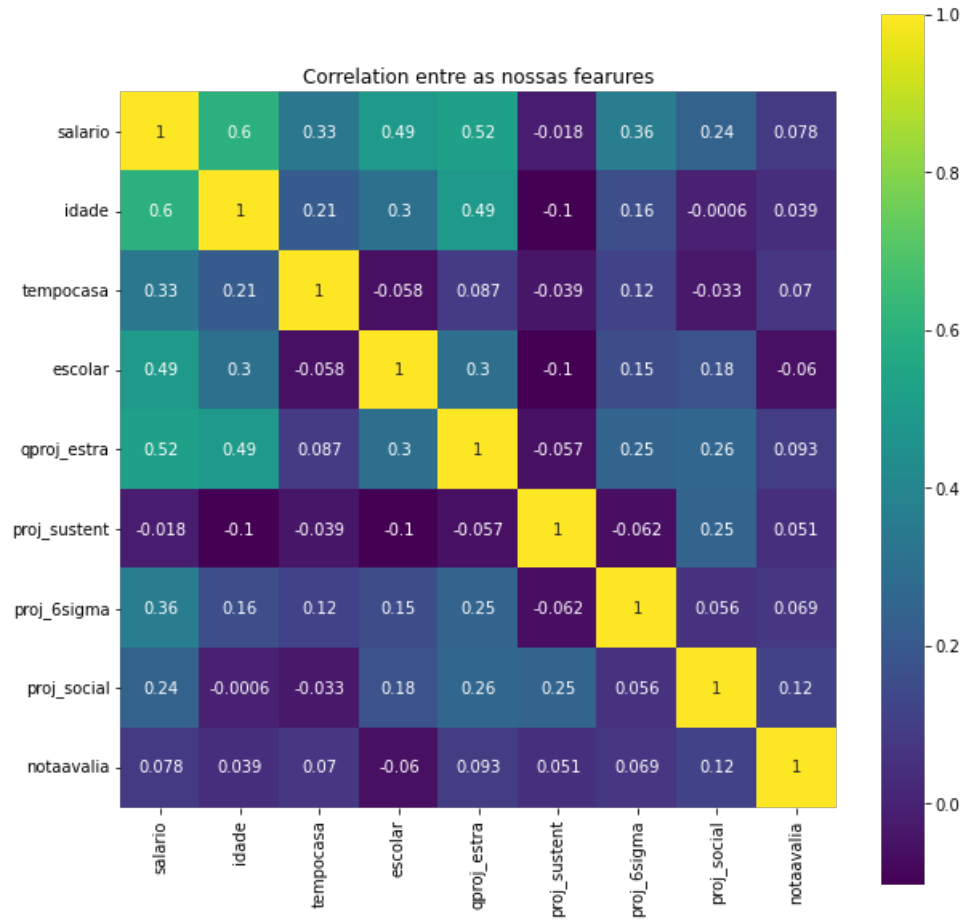
	salario	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavalia
salario	1.000000	0.600277	0.332923	0.492168	0.518249	-0.017883	0.359549	0.237473	0.078258
idade	0.600277	1.000000	0.212797	0.299943	0.488883	-0.103279	0.161561	-0.000596	0.039255
tempocasa	0.332923	0.212797	1.000000	-0.058369	0.087035	-0.039051	0.115490	-0.032775	0.069828
escolar	0.492168	0.299943	-0.058369	1.000000	0.297009	-0.102464	0.153629	0.175625	-0.059922
qproj_estra	0.518249	0.488883	0.087035	0.297009	1.000000	-0.057288	0.248645	0.260660	0.093075
proj_sustent	-0.017883	-0.103279	-0.039051	-0.102464	-0.057288	1.000000	-0.062330	0.248069	0.051128
proj_6sigma	0.359549	0.161561	0.115490	0.153629	0.248645	-0.062330	1.000000	0.055835	0.069299
proj_social	0.237473	-0.000596	-0.032775	0.175625	0.260660	0.248069	0.055835	1.000000	0.115789
notaavalia	0.078258	0.039255	0.069828	-0.059922	0.093075	0.051128	0.069299	0.115789	1.000000

In [151]:

```
correlation = salariosBase.corr()
plt.figure(figsize=(10,10))
sb.heatmap(correlation, vmax=1, square=True,annot=True,cmap='viridis')

plt.title('Correlation entre as nossas feaures')

plt.show()
```



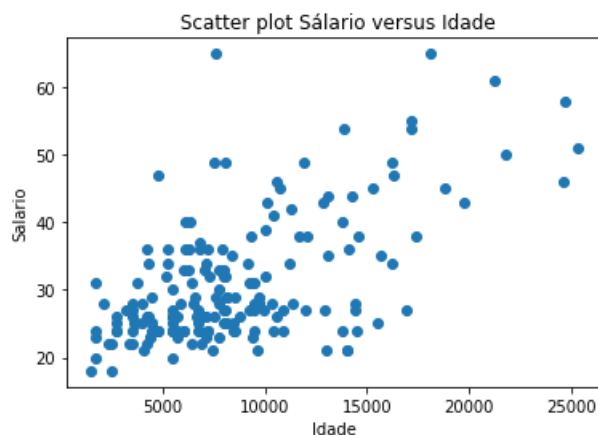
Nossa variável target é salário, sendo assim realizamos um scatter plot do salário x a idade para verificar a relação de peso da idade do funcionário com o salário.

In [152]:

```
plt.scatter('salario', 'idade', data=salariosBase)
plt.xlabel("Idade")
plt.ylabel("Salario")
plt.title(" Scatter plot Sálario versus Idade")
```

Out[152]:

Text(0.5, 1.0, ' Scatter plot Sálario versus Idade')



Análise das variáveis preditoras/independentes

Realizamos uma análise das outras variáveis x a nossa variável target, essa análise foi feita nas variáveis:

- salario x tempocasa
- salario x escolar
- salario x qproj_estra
- salario x proj_sustent
- salario x proj_6sigma
- salario x proj_social
- salario x notaavalia

In [153]:

```
plt.figure(figsize=(24,20))
```

```
plt.subplot(4, 2, 1)
plt.scatter(salariosBase['tempocasa'], salariosBase['salario'], marker='o', color='g');
fig.set_title('Análise salário x tempo de casa')
fig.set_ylabel('Sálario')
fig.set_xlabel('Tempo de casa')

plt.subplot(4, 2, 2)
plt.scatter(salariosBase['escolar'], salariosBase['salario'], marker='o', color='b');
fig.set_title('Análise salário x escolaridade')
fig.set_ylabel('Sálario')
fig.set_xlabel('Escolaridade')

plt.subplot(4, 2, 3)
plt.scatter(salariosBase['qproj_estra'], salariosBase['salario'], marker='o', color='y');
fig.set_title('Análise salário x projetos estratégicos')
fig.set_ylabel('Sálario')
fig.set_xlabel('Projetos estratégicos')

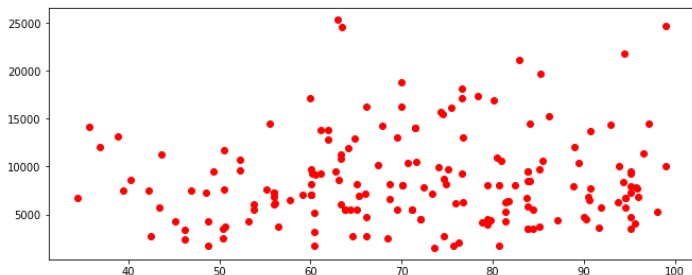
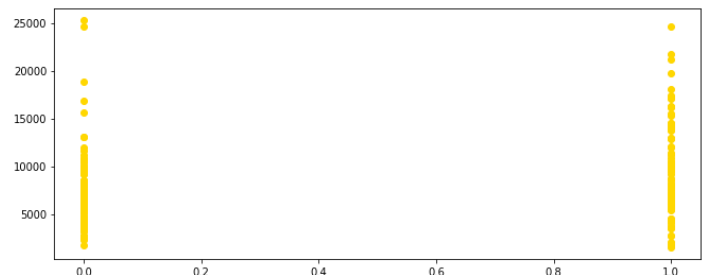
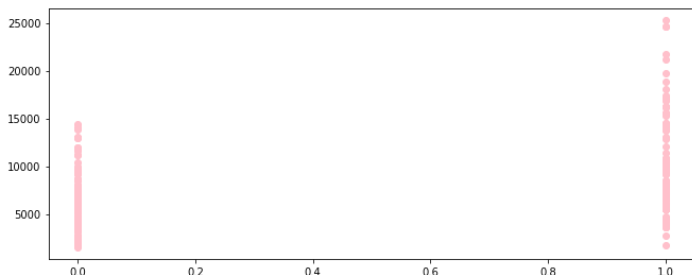
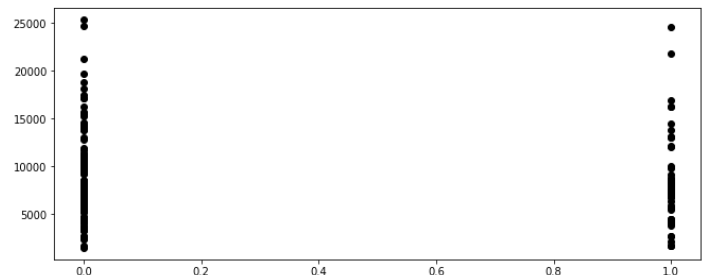
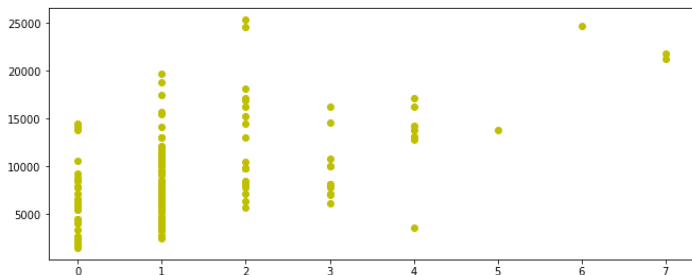
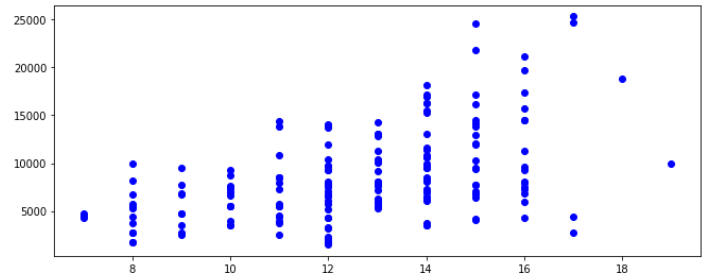
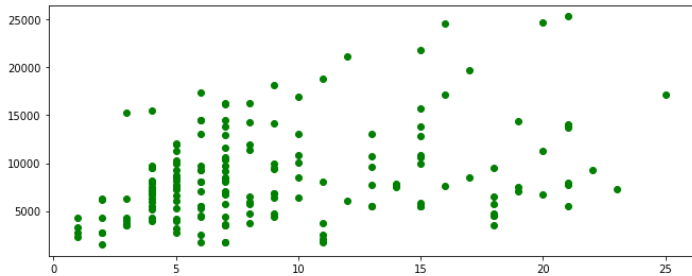
plt.subplot(4, 2, 4)
plt.scatter(salariosBase['proj_sustent'], salariosBase['salario'], marker='o', color='black');
fig.set_title('Análise salário x projetos sustentáveis')
fig.set_ylabel('Sálario')
fig.set_xlabel('Projetos sustentáveis')

plt.subplot(4, 2, 5)
plt.scatter(salariosBase['proj_6sigma'], salariosBase['salario'], marker='o', color='pink');
fig.set_title('Análise salário x projetos padrão 6Sigma')
fig.set_ylabel('Sálario')
fig.set_xlabel('Projetos padrão 6Sigma')
```

```
plt.subplot(4, 2, 6)
plt.scatter(salariosBase['proj_social'], salariosBase['salario'], marker='o', color='gold');
fig.set_title('Análise salário x projetos social')
fig.set_ylabel('Salário')
fig.set_xlabel('Projetos social')

plt.subplot(4, 2, 7)
plt.scatter(salariosBase['notaavalia'], salariosBase['salario'], marker='o', color='red');
fig.set_title('Análise salário x nota de avaliação')
fig.set_ylabel('Salário')
fig.set_xlabel('nota de avaliação')

plt.show()
```



Boxplot variável target x variáveis qualitativa

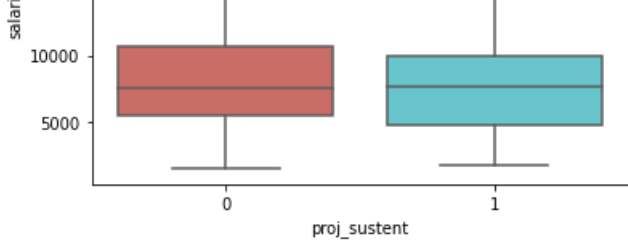
Realizamos uma análise das variáveis qualitativas x a nossa variável target, essa análise foi feita usando as seguintes variáveis:

- salario x proj_sustent
- salario x proj_6sigma
- salario x proj_social

In [154]:

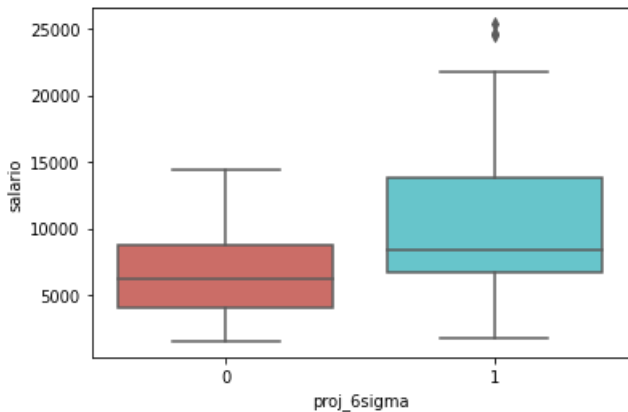
```
sb.boxplot(x='proj_sustent', y='salario', data=salariosBase, palette='hls');
```





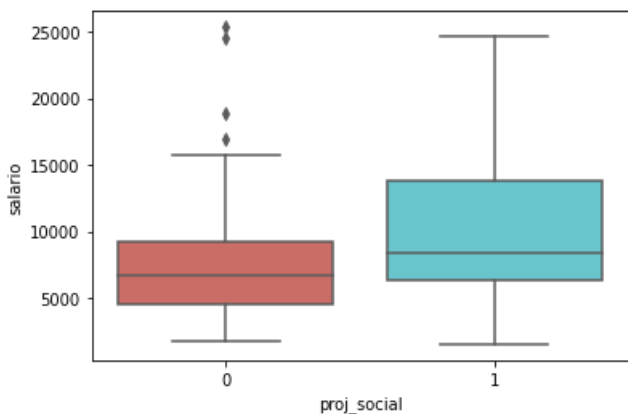
In [155]:

```
sb.boxplot(x='proj_6sigma', y='salario', data=salariosBase, palette='hls');
```



In [156]:

```
sb.boxplot(x='proj_social', y='salario', data=salariosBase, palette='hls');
```



Criação do teste qui-quadrado

Faremos isso para descobrir a associação entre as variáveis qualitativas.

Transformamos a variável salário em qualitativa com o nome de fx_salario e testaremos com:

- salario x proj_sustent
- salario x proj_6sigma
- salario x proj_social

In [157]:

```
Q1 = np.percentile(salariosBase['salario'], 25)
Q2 = np.percentile(salariosBase['salario'], 50)
Q3 = np.percentile(salariosBase['salario'], 75)
maximo = max(salariosBase['salario'])
print(Q1, Q2, Q3, maximo)
```

5491.23000000000005 7551.125 10584.685 25329.91

In [158]:

```
salariosBase.loc[(salariosBase.salario >= 0) & (salariosBase.salario <= Q1), 'fx_salario']=1.0
salariosBase.loc[(salariosBase.salario > Q1) & (salariosBase.salario <= Q2), 'fx_salario']=2.0
```

```
salariosBase.loc[(salariosBase.salario > Q2) & (salariosBase.salario <= Q3), 'fx_salario']=3.0
salariosBase.loc[(salariosBase.salario > Q3) & (salariosBase.salario <= maximo), 'fx_salario']=4.0
```

In [159]:

```
salariosBase.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 10 columns):
#   Column      Non-Null Count  Dtype
---  -
0   salario     180 non-null   float64
1   idade       180 non-null   int64
2   tempocasa   180 non-null   int64
3   escolar     180 non-null   int64
4   qproj_estra 180 non-null   int64
5   proj_sustent 180 non-null   int64
6   proj_6sigma 180 non-null   int64
7   proj_social 180 non-null   int64
8   notaavalia  180 non-null   float64
9   fx_salario  180 non-null   float64
dtypes: float64(3), int64(7)
memory usage: 14.2 KB
```

In [160]:

```
salariosBase.head()
```

Out[160]:

	salario	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavalia	fx_salario
0	8000.80	25	4	11	1	1	1	0	79.38	3.0
1	8500.17	24	5	11	0	0	1	0	84.13	3.0
2	3350.59	22	1	12	0	0	0	0	46.15	1.0
3	9500.24	28	4	14	1	0	0	1	83.85	3.0
4	1500.63	18	2	12	0	0	0	1	73.64	1.0

Realizando o crosstab

Agora vamos realizar o crosstab das variáveis:

- salario x proj_sustent
- salario x proj_6sigma
- salario x proj_social

Crosstab fx_salario x proj_sustent

In [161]:

```
pd.crosstab(salariosBase.fx_salario, salariosBase.proj_sustent, margins=True)
```

Out[161]:

proj_sustent	0	1	All
fx_salario			
1.0	32	13	45
2.0	34	11	45
3.0	31	14	45
4.0	33	12	45
All	130	50	180

In [162]:

```
(salariosBase.proj_sustent.value_counts() / salariosBase.shape[0]) * 100
```

Out[162]:

```
0    72.222222
1    27.777778
Name: proj_sustent, dtype: float64
```

In [163]:

```
table_proj_sustent = pd.crosstab(salariosBase.fx_salario,salariosBase.proj_sustent)
```

In [164]:

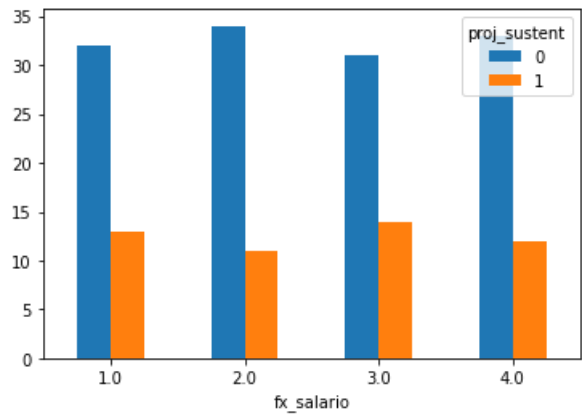
```
table_proj_sustent
```

Out[164]:

proj_sustent	0	1
fx_salario		
1.0	32	13
2.0	34	11
3.0	31	14
4.0	33	12

In [165]:

```
barplot_proj_sustent = table_proj_sustent.plot.bar(rot=0)
```



Crosstab fx_salario x proj_6sigma

In [166]:

```
pd.crosstab(salariosBase.fx_salario, salariosBase.proj_6sigma, margins=True)
```

Out[166]:

proj_6sigma	0	1	All
fx_salario			
1.0	31	14	45
2.0	23	22	45
3.0	16	29	45
4.0	11	34	45
All	81	99	180

In [167]:

```
(salariosBase.proj_6sigma.value_counts() / salariosBase.shape[0]) * 100
```

Out[167]:

```
1    55.0
```

0 35.0
0 45.0
Name: proj_6sigma, dtype: float64

In [168]:

```
table_proj_6sigma = pd.crosstab(salariosBase.fx_salario,salariosBase.proj_6sigma)
```

In [169]:

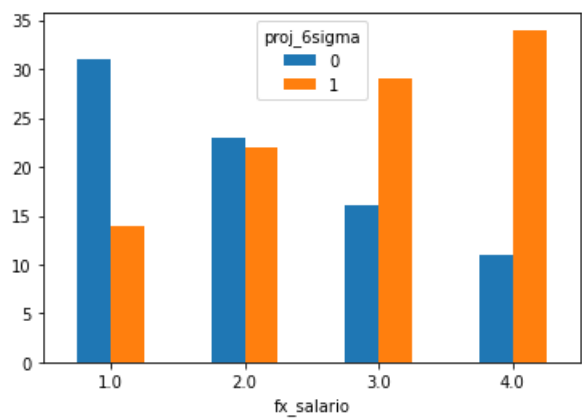
```
table_proj_6sigma
```

Out[169]:

proj_6sigma	0	1
fx_salario		
1.0	31	14
2.0	23	22
3.0	16	29
4.0	11	34

In [170]:

```
barplot_proj_6sigma = table_proj_6sigma.plot.bar(rot=0)
```



Crosstab fx_salario x proj_social

In [171]:

```
pd.crosstab(salariosBase.fx_salario, salariosBase.proj_social, margins=True)
```

Out[171]:

proj_social	0	1	All
fx_salario			
1.0	29	16	45
2.0	26	19	45
3.0	22	23	45
4.0	13	32	45
All	90	90	180

In [172]:

```
(salariosBase.proj_social.value_counts() / salariosBase.shape[0]) * 100
```

Out[172]:

0 50.0
1 50.0
Name: proj_social, dtype: float64

In [173]:

```
table_proj_social = pd.crosstab(salariosBase.fx_salario,salariosBase.proj_social)
```

In [174]:

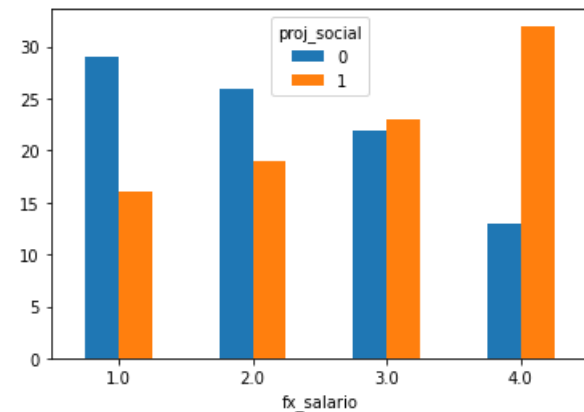
```
table_proj_social
```

Out[174]:

proj_social	0	1
fx_salario		
1.0	29	16
2.0	26	19
3.0	22	23
4.0	13	32

In [175]:

```
barplot_proj_social = table_proj_social.plot.bar(rot=0)
```



Utilização da função chi2_contingency

Nós utilizaremos essa função para calcular a estatística qui-quadrado e o p-value para o teste de hipótese de independência das frequências observadas na tabela de contingências:

- table_proj_sustent
- table_proj_6sigma
- table_proj_social

Teste de hipótese

- H0: não há relação entre salario e as variaveis que serão testadas
- H1: há relação entre salario e as variaveis que serão testadas

$\alpha = 0,05$ (5%)

Função testChi2

Criada para evitar a repetição da instrução de teste.

In [176]:

```
def testChi2(dof, expected, stat, p):  
    print('Graus de liberdade: dof=%d' % dof)  
  
    print("\n Tabela Esperada:")  
    print(expected)  
  
    # interpret test-statistic  
    prob = 0.95  
    critical = chi2.ppf(prob, dof)
```



```
print("\n probability=%.3f, critical=%.3f, stat=%.3f" % (prob, critical, stat))
```

```
if abs(stat) >= critical:
    print("\n Dependente (rejeita H0)")
else:
    print("\n Independente (falha em rejeitar H0)")
# interpret p-value
alpha = 1.0 - prob
print("\n significance=%.3f, p=%.3f" % (alpha, p))
if p <= alpha:
    print("\n Dependente (rejeita H0)")
else:
    print("\n Independente (falha em rejeitar H0)")
```

chi2_contingency para table_proj_sustent

In [177]:

```
stat_proj_sustent, p_proj_sustent, dof_proj_sustent, expected_proj_sustent = chi2_contingency(table_proj_sustent)
p_proj_sustent
```

Out[177]:

0.9069120268597668

In [178]:

```
testChi2(dof=dof_proj_sustent, expected=expected_proj_sustent, stat=stat_proj_sustent, p=p_proj_sustent)
```

Graus de liberdade: dof=3

Tabela Esperada:

```
[[32.5 12.5]
 [32.5 12.5]
 [32.5 12.5]
 [32.5 12.5]]
```

probability=0.950, critical=7.815, stat=0.554

Independente (falha em rejeitar H0)

significance=0.050, p=0.907

Independente (falha em rejeitar H0)

chi2_contingency para table_proj_6sigma

In [179]:

```
stat_proj_6sigma, p_proj_6sigma, dof_proj_6sigma, expected_proj_6sigma = chi2_contingency(table_proj_6sigma)
p_proj_6sigma
```

Out[179]:

0.0001429985968152404

In [180]:

```
testChi2(dof=dof_proj_6sigma, expected=expected_proj_6sigma, stat=stat_proj_6sigma, p=p_proj_6sigma)
```

Graus de liberdade: dof=3

Tabela Esperada:

```
[[20.25 24.75]
 [20.25 24.75]
 [20.25 24.75]
 [20.25 24.75]]
```

probability=0.950, critical=7.815, stat=20.359

Dependente (rejeita H0)

significance=0.050, p=0.000

Dependente (rejeita H0)

chi2_contingency para table_proj_social

In [181]:

```
stat_proj_social, p_proj_social, dof_proj_social, expected_proj_social = chi2_contingency(table_proj_social)
p_proj_social
```

Out[181]:

0.004883155862460387

In [182]:

```
testChi2(dof_proj_social, expected_proj_social, stat_proj_social, p_proj_social)
```

Graus de liberdade: dof=3

Tabela Esperada:

[[22.5 22.5]

[22.5 22.5]

[22.5 22.5]

[22.5 22.5]]

probability=0.950, critical=7.815, stat=12.889

Dependente (rejeita H0)

significance=0.050, p=0.005

Dependente (rejeita H0)

Execução do teste utilizando researchpy

Aqui nós vamos executar a função crosstab utilizando o teste chi-square que irá realizar o teste qui-quadrado das nossas variáveis:

- fx_salario x proj_sustent
- fx_salario x proj_6sigma
- fx_salario x proj_social

In [183]:

```
resultado_fx_salario_proj_sustent = researchpy.crosstab(salariosBase['fx_salario'], salariosBase['proj_sustent'], test='chi-square')
```

In [184]:

```
resultado_fx_salario_proj_sustent
```

Out[184]:

```
(      proj_sustent
proj_sustent    0  1 All
fx_salario
1.0           32 13 45
2.0           34 11 45
3.0           31 14 45
4.0           33 12 45
All           130 50 180,      Chi-square test  results
0 Pearson Chi-square ( 3.0) =  0.5538
1      p-value =  0.9069
2      Cramer's V =  0.0555)
```

In [185]:

```
resultado_fx_salario_proj_6sigma = researchpy.crosstab(salariosBase['fx_salario'], salariosBase['proj_6sigma'], test='chi-square')
```

In [186]:

```
resultado_fx_salario_proj_6sigma
```

Out[186]:

Out[186]:

```
(      proj_6sigma
proj_6sigma    0  1  All
fx_salario
1.0          31 14  45
2.0          23 22  45
3.0          16 29  45
4.0          11 34  45
All          81 99 180,      Chi-square test results
0 Pearson Chi-square ( 3.0) = 20.3591
1          p-value = 0.0001
2          Cramer's V = 0.3363)
```

In [187]:

```
resultado_fx_salario_proj_social = researchpy.crosstab(salariosBase['fx_salario'], salariosBase['proj_social'], test='chi-square')
```

In [188]:

```
resultado_fx_salario_proj_social
```

Out[188]:

```
(      proj_social
proj_social    0  1  All
fx_salario
1.0          29 16  45
2.0          26 19  45
3.0          22 23  45
4.0          13 32  45
All          90 90 180,      Chi-square test results
0 Pearson Chi-square ( 3.0) = 12.8889
1          p-value = 0.0049
2          Cramer's V = 0.2676)
```

Início do nosso teste utilizando o algoritmo de regressão linear

Agora que já realizamos a análise exploratória, verificamos todos os dados podemos então rodar a nossa regressão linear que irá verificar o relacionamento das nossas variáveis.

Atribuição da função para o modelo utilizando regressão linear

Também verificamos o coeficiente para rodar o nosso modelo

In [189]:

```
lm = LinearRegression()
```

In [190]:

```
x = salariosBase[['idade', 'tempocasa', 'escolar', 'qproj_estra', 'proj_sustent', 'proj_6sigma', 'proj_social', 'notaavaliao']]
y = salariosBase[['salario']]
```

In [191]:

```
x
```

Out[191]:

	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavaliao
0	25	4	11	1	1	1	0	79.38
1	24	5	11	0	0	1	0	84.13
2	22	1	12	0	0	0	0	46.15
3	28	4	14	1	0	0	1	83.85
4	18	2	12	0	0	0	1	73.64
...
175	36	14	13	0	1	1	1	88.87
176	26	15	14	1	0	1	1	85.45

177	idade36	tempocasa3	escolar14	qproj_estra1	proj_sustent0	proj_6sigma0	proj_social1	notaavalia76.72
178	28	11	12	0	1	0	1	76.22
179	21	3	15	1	1	0	1	79.40

180 rows × 8 columns

In [192]:

```
y
```

Out[192]:

	salario
0	8000.80
1	8500.17
2	3350.59
3	9500.24
4	1500.63
...	...
175	7896.70
176	10575.13
177	6309.66
178	2100.68
179	4059.13

180 rows × 1 columns

Iniciando o processo de treino do nosso modelo

In [193]:

```
lm.fit(x,y)
```

Out[193]:

LinearRegression()

In [194]:

```
# Termo independente no modelo linear
lm.intercept_
```

Out[194]:

array([-9011.78914327])

In [195]:

```
# Coeficientes estimados para o nosso problema de regressão linear
lm.coef_
```

Out[195]:

array([[179.31785454, 207.14917154, 571.52233188, 596.95436422,
 503.25814193, 1729.11169862, 1193.86954237, 6.43057797]])

Visualizando os coeficientes

A seguir nos verificaremos os coeficientes das variáveis testadas para o nosso modelo.

In [196]:

```
coeficientes = pd.concat([pd.DataFrame(x.columns),pd.DataFrame(np.transpose(lm.coef_))], axis = 1)
```

In [197]:

coeficientes.T

Out[197]:

	0	1	2	3	4	5	6	7
0	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavalia
0	179.317855	207.149172	571.522332	596.954364	503.258142	1729.111699	1193.869542	6.430578

In [198]:

```
X = salariosBase[salariosBase.columns[1:9]]  
  
y = salariosBase[['salario']]
```

In [199]:

```
total_amostras = 101  
tamanho_do_teste = 0.05
```

Criando nossa base de testes e treino

Utilizaremos a função train_test_split que irá separar nosso dataset em teste e treino.

In [200]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = tamanho_do_teste, random_state = total_amostras)
```

Visualização do resultado com Mínimos Quadrados Ordinários

Nós utilizamos a técnica de Mínimos Quadrados Ordinários que consiste em uma técnica de otimização matemática que procura encontrar o melhor ajuste para um conjunto de dados tentando minimizar a soma dos quadrados das diferenças entre o valor estimado e os dados observados.

In [201]:

```
X_ = sm.add_constant(X_train)  
  
model = sm.OLS(y_train, X_).fit()  
  
print(model.summary())
```

OLS Regression Results						
=====						
Dep. Variable:	salario	R-squared:	0.618			
Model:	OLS	Adj. R-squared:	0.600			
Method:	Least Squares	F-statistic:	32.83			
Date:	Fri, 16 Sep 2022	Prob (F-statistic):	2.92e-30			
Time:	21:53:01	Log-Likelihood:	-1605.0			
No. Observations:	171	AIC:	3228.			
Df Residuals:	162	BIC:	3256.			
Df Model:	8					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

const	-9223.5683	1713.319	-5.383	0.000	-1.26e+04	-5840.250
idade	179.8166	29.312	6.134	0.000	121.933	237.700
tempocasa	196.9574	41.860	4.705	0.000	114.296	279.618
escolar	601.9550	98.534	6.109	0.000	407.378	796.532
qproj_estra	597.7543	236.873	2.524	0.013	129.998	1065.510
proj_sustent	597.9183	529.065	1.130	0.260	-446.836	1642.672
proj_6sigma	1779.4345	473.560	3.758	0.000	844.287	2714.582
proj_social	1024.1501	497.210	2.060	0.041	42.302	2005.998
notaavalia	6.5617	14.471	0.453	0.651	-22.014	35.138
=====						
Omnibus:	3.153	Durbin-Watson:	2.031			
Prob(Omnibus):	0.207	Jarque-Bera (JB):	2.696			
Skew:	0.277	Prob(JB):	0.260			
Kurtosis:	3.269	Cond. No.	615.			
=====						

Notes:

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a future version of pandas all arguments of concat except f
or the argument 'objs' will be keyword-only
x = pd.concat(x[::order], 1)
```

R2 da regressão

No caso anterior no **R2=0.600**, neste caso ficou em **60%** após avaliarmos isso decidimos então alterar as variáveis significantes e rodar o modelo novamente.

In [202]:

```
X_train, X_test, y_train, y_test = train_test_split(salariosBase.drop(['salario', 'notaavalia'], axis=1), salariosBase['salario'], test_size=tamanho_do_teste, random_state=17)
```

In [203]:

```
X_ = sm.add_constant(X_train)
modelo = sm.OLS(y_train, X_).fit()
print(modelo.summary())
```

OLS Regression Results

```
=====
Dep. Variable:          salario  R-squared:          0.840
Model:                  OLS      Adj. R-squared:      0.832
Method:                 Least Squares  F-statistic:    106.0
Date:                   Fri, 16 Sep 2022  Prob (F-statistic): 2.43e-60
Time:                   21:53:01  Log-Likelihood:    -1532.2
No. Observations:       171  AIC:                   3082.
Df Residuals:           162  BIC:                   3111.
Df Model:                8
Covariance Type:        nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	-5356.1192	909.582	-5.889	0.000	-7152.284	-3559.954
idade	82.0865	20.009	4.102	0.000	42.574	121.599
tempocasa	87.1435	28.955	3.010	0.003	29.965	144.322
escolar	207.3210	68.544	3.025	0.003	71.966	342.677
qproj_estra	414.3179	153.770	2.694	0.008	110.666	717.970
proj_sustent	158.9158	346.483	0.459	0.647	-525.290	843.122
proj_6sigma	537.5852	322.239	1.668	0.097	-98.746	1173.916
proj_social	137.6092	331.049	0.416	0.678	-516.119	791.337
fx_salario	2780.5329	182.768	15.213	0.000	2419.618	3141.448

```
=====
Omnibus:                57.351  Durbin-Watson:          2.235
Prob(Omnibus):           0.000  Jarque-Bera (JB):      213.852
Skew:                    1.248  Prob(JB):              3.65e-47
Kurtosis:                7.877  Cond. No.              223.
=====
```

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a future version of pandas all arguments of concat except f
or the argument 'objs' will be keyword-only
x = pd.concat(x[::order], 1)
```

Ajuste do modelo

Agora que ajustamos o nosso treino e teste nosso **R2=0.832**, neste caso ficando então em **83%**

Medidas de erro

Agora para testar nós utilizamos duas medidas de erro e são elas:

- mean_absolute_error
- mean_squared_error

Onde mean_absolute_error que é o erro absoluto médio é uma medida de erros entre observações pareadas que expressam o mesmo fenômeno.

E `mean_squared_error` irá retornar o nosso erro quadrático médio ou risco quadrático de um estimador.

Vamos então agora calcular o valor predito da nossa variável de reposta que é retornada da nossa amostrar de treino.

In [204]:

```
y_train_pred = modelo.predict(X_)
```

In [205]:

```
y_train_pred
```

Out[205]:

```
0      8777.565257
90     10470.862083
11      5795.415518
172    14978.489561
73     15630.522214
...
57      3390.786433
150     3667.588207
134     6072.017259
143    12812.114773
111    13167.159182
Length: 171, dtype: float64
```

Ajustando as medidas de erro da amostra

In [206]:

```
me1 = round((y_train-y_train_pred).mean(),2)
mae1 = (mean_absolute_error(y_train, y_train_pred)).round(2)
mse1 = (mean_squared_error(y_train, y_train_pred)).round(2)
rmse1 = (np.sqrt(mean_squared_error(y_train, y_train_pred))).round(2)
mpe1 = round(((y_train - y_train_pred)/y_train).mean(),2)
mape1 = round((mae1/y_train).mean(),2)
```

In [207]:

```
lista_com_as_medidas = [me1, mae1,mse1,rmse1,mpe1, mape1]

pd.DataFrame({"treino_resultado": lista_com_as_medidas})
```

Out[207]:

treino_resultado	
0	-0.00
1	1321.36
2	3551146.93
3	1884.45
4	-0.02
5	0.21

Gráficos de Diagnóstico dos Resíduos

Agora nós faremos a parte visual e verificar nosso Residuals vs Fitted essa informações indicam a existência de padrões não lineares nos resíduos.

Explicações a cerca dos itens:

- Normal Q-Q - examina se os resíduos são normalmente distribuídos. É bom que os pontos residuais sigam a linha reta.
- Homoscedasticidade - verifica a homogeneidade de variância dos resíduos. A linha horizontal com pontos igualmente dispersos é uma boa indicação de homoscedasticidade.
- Residuals vs Leverage - identifica casos influentes, ou seja, valores extremos que podem influenciar os resultados da regressão quando incluídos ou excluídos da análise.
- Antes de se construir os gráficos, em primeiro construir-se-á o modelo de resíduo para cada diagnóstico.

Criaremos então a seguir os diagnósticos para então plotar os gráficos

In [208]:

```
modelo_ajust_y = modelo.fittedvalues
modelo_residuos = modelo.resid
modelo_norm_resid = modelo.get_influence().resid_studentized_internal
modelo_norm_resid_abs = np.sqrt(np.abs(modelo_norm_resid))
modelo_outliers = modelo.get_influence().hat_matrix_diag
distancia_cook = modelo.get_influence().cooks_distance[0]
```

In [209]:

```
modelo_ajust_y.head()
```

Out[209]:

```
0      8777.565257
90     10470.862083
11      5795.415518
172     14978.489561
73     15630.522214
dtype: float64
```

In [210]:

```
modelo_norm_resid
```

Out[210]:

```
array([-0.41372582, -1.30365394, -0.15474994, -0.62682323, -1.47671575,
       -0.83119167, -0.26841235, -0.77246125, -0.72631581,  0.61306468,
        0.81793179,  0.96393368, -0.84368306,  0.03930418, -0.38828879,
        3.18165483, -1.19203684,  0.08064021, -0.12666786,  2.14024497,
        0.62641817,  0.07609753, -0.26694983, -2.86629919,  0.49712426,
       -0.46159803,  0.20869222, -0.06176111,  0.06487283, -1.16506695,
        0.34520695,  0.04563745,  1.97291097, -0.10749237, -0.05311782,
        0.1161306 , -1.48658701,  4.64269477, -0.92989752, -1.04677052,
       -1.04505439, -0.95508077,  0.18510345, -1.03357434, -0.25800735,
        0.13761973,  1.2067251 ,  0.93459222,  0.0471428 , -0.17451346,
        1.18894671,  0.44140174, -0.19310928,  0.0761631 ,  1.92980264,
       -0.48216906, -0.90133404, -0.49653719,  1.14421218,  0.76524478,
       -0.33308621, -0.1771622 , -0.29851955, -0.75126926, -0.31046609,
       -1.72054975, -0.06727465, -0.34119934, -0.33141177,  0.68311375,
       -0.29235461,  0.32359383, -1.10130273, -1.59315345,  1.90170545,
       -1.877028  , -0.46117855, -0.07281963,  0.56711951,  0.28659393,
       -1.20983641, -1.15605013, -0.48581843, -0.43202389,  0.26097411,
        0.06633036, -0.10627302, -0.9053784 ,  0.15368431, -0.13994558,
       -0.19201608, -1.22488047,  0.2937315 , -0.53459883,  0.38594224,
        1.00563196, -0.11083857, -0.08906039, -1.07051332,  0.19410988,
        0.52834439, -0.86636717,  4.80645179,  0.86782007,  0.01846646,
       -0.34978914,  0.03258479,  0.74674696, -1.03944157,  0.84507186,
        0.21736435,  1.1518153 ,  0.38886835,  0.81825526, -2.37898626,
       -0.29012878, -0.8286432 , -0.44953097, -0.02063869,  0.00784865,
        0.7742894 , -0.7256796 , -0.44171673, -0.0721979 , -1.42223021,
        1.20357925,  0.55519737, -0.70776021,  0.95069613, -0.19721321,
        0.30983525,  1.41170937, -0.37187608, -1.1074557 ,  0.08212781,
        0.96991138, -0.3572697 ,  0.44228926, -0.30180625, -0.53071782,
       -0.13383115,  0.99557166,  0.41696725, -0.85120609,  2.40170194,
        0.16279586,  0.81202214, -1.00232426,  0.24545725,  0.75132641,
       -0.14779661, -0.45514604, -0.33356724,  0.22393817,  0.05416107,
        0.46468069,  0.43876644, -0.02511206, -0.26727797,  0.17306349,
        0.4648694 ,  1.33714202, -0.93309237,  1.76903896, -0.70658941,
       -1.7081736 ,  0.58973881,  0.02103974,  0.74518748, -0.61085494,
        0.50953213])
```

Criação de um dataframe para visualização dos dados de salário

Criamos um dataframe para então verificar a variável de salário x as observadas pelo nosso modelo

In [211]:

```
data_salario_x_modelo = pd.concat([y_train, modelo_ajust_y], axis=1)
```

In [212]:

```
data_salario_x_modelo.head()
```


Out[212]:

	salario	0
0	8000.80	8777.565257
90	8025.24	10470.862083
11	5500.80	5795.415518
172	13827.91	14978.489561
73	12843.63	15630.522214

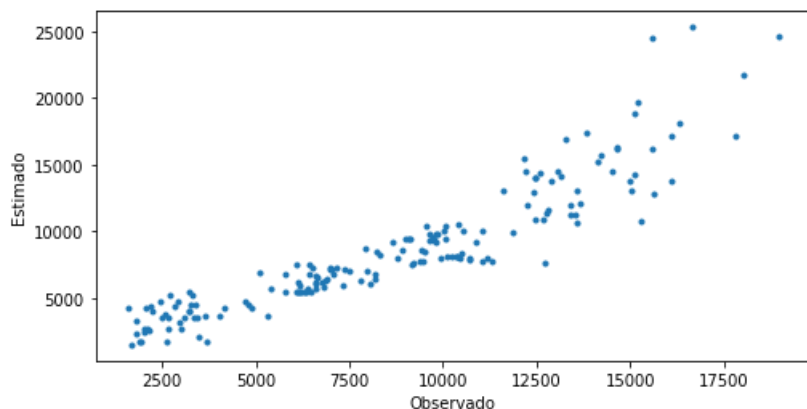
Plot do gráfico observado x estimado

In [213]:

```
plt.figure(figsize=(8, 4))
plt.scatter(y_train_pred, y_train, marker='.')
plt.xlabel("Observado")
plt.ylabel("Estimado")
```

Out[213]:

Text(0, 0.5, 'Estimado')



Verificação de padrão nos resíduos

Aqui executamos um plot para verificar se existe algum problema em nosso modelo linear, este método é usado para podermos representar graficamente os resíduos da regressão linear e com isso definirmos visualmente a suposição da nossa normalidade, porém ainda não mostra se podemos aceitar os resíduos como normais.

In [214]:

```
plot_lm_1 = plt.figure()
plot_lm_1.axes[0] = sb.residplot(modelo_ajust_y, data_salario_x_modelo.columns[-0], data=data_salario_x_modelo, scatter_kws={'alpha': 0.5}, line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})

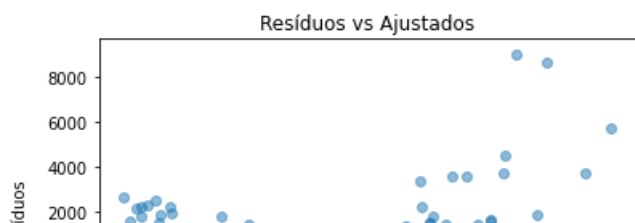
plot_lm_1.axes[0].set_title('Resíduos vs Ajustados')
plot_lm_1.axes[0].set_xlabel('Valores ajustados')
plot_lm_1.axes[0].set_ylabel('Resíduos')
```

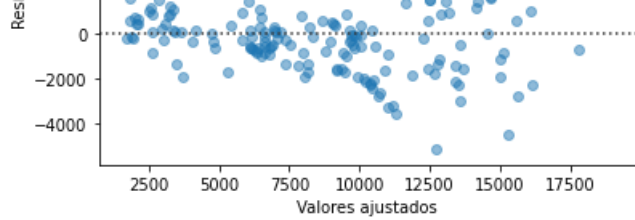
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation

FutureWarning

Out[214]:

Text(0, 0.5, 'Resíduos')





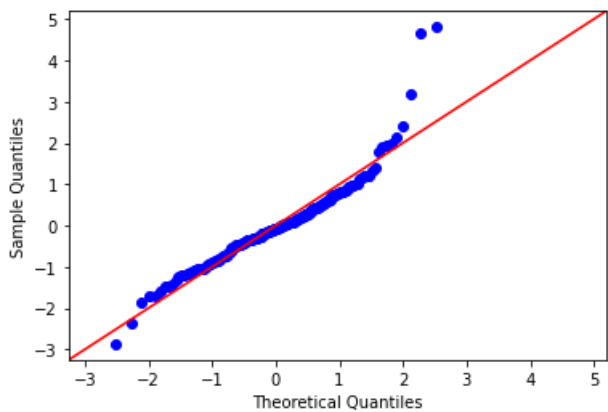
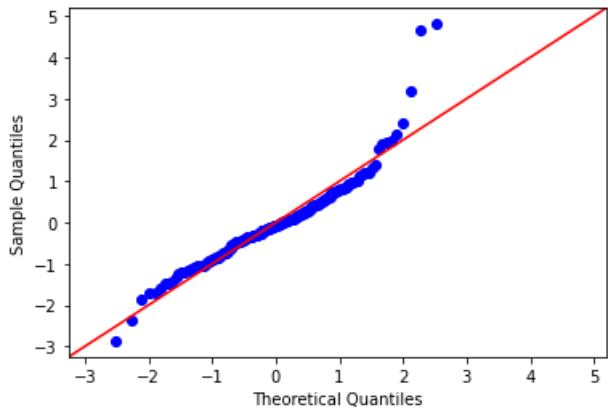
Teste de normalidade

Plot de gráfico para verificar se nossos resíduos estão normalmente distribuídos

In [215]:

```
sm.qqplot(modelo_norm_resid, line='45')
```

Out[215]:



Shapiro-Wilk

In [216]:

```
shapiro_test = stats.shapiro(modelo_norm_resid)
shapiro_test
```

Out[216]:

ShapiroResult(statistic=0.9174700379371643, pvalue=2.9718696836766867e-08)

Teste de Homoscedasticidade

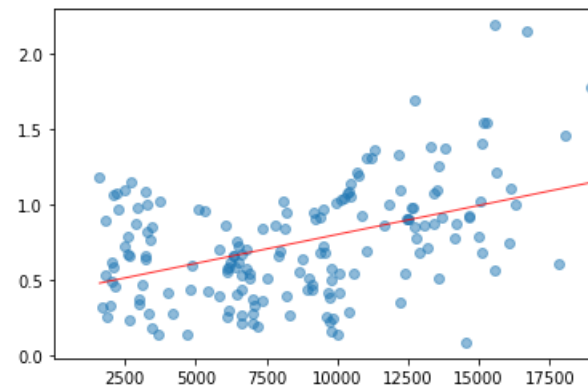
In [217]:

```
plot_lm_3 = plt.figure()
plt.scatter(modelo_ajust_y, modelo_norm_resid_abs, alpha=0.5);
sb.regplot(modelo_ajust_y, modelo_norm_resid_abs,
            scatter=False,
            ci=False,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8});
plot_lm_3.axes[0].set_xlim(4, max(modelo_ajust_y)+0.05)
```

```
plt.show()
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation

FutureWarning



Pontos influentes e valores extremos

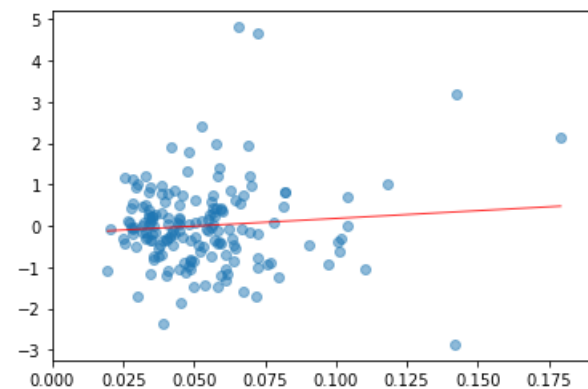
In [218]:

```
plot_lm_4 = plt.figure();
plt.scatter(modelo_outliers, modelo_norm_resid, alpha=0.5);
sb.regplot(modelo_outliers, modelo_norm_resid,
            scatter=False,
            ci=False,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8});
plot_lm_4.axes[0].set_xlim(0, max(modelo_outliers)+0.01)

plt.show()
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation

FutureWarning



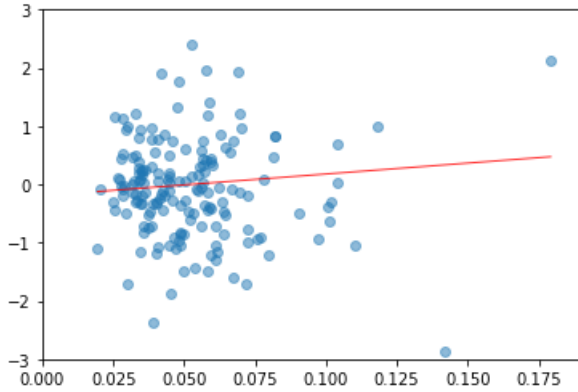
In [219]:

```
plot_lm_4 = plt.figure();
plt.scatter(modelo_outliers, modelo_norm_resid, alpha=0.5);
sb.regplot(modelo_outliers, modelo_norm_resid,
            scatter=False,
            ci=False,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8});
plot_lm_4.axes[0].set_xlim(0, max(modelo_outliers)+0.01);
plot_lm_4.axes[0].set_ylim(-3, 3)

plt.show()
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation

FutureWarning



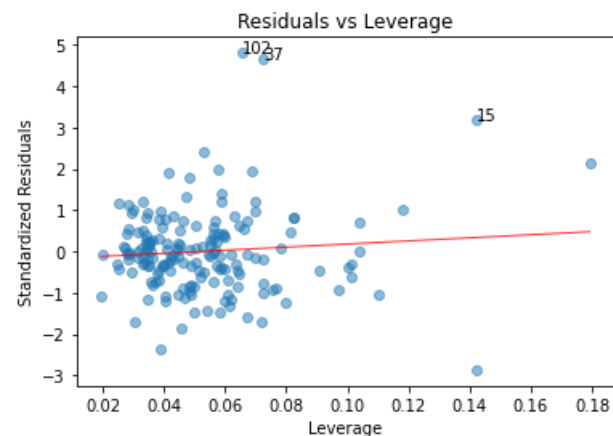
In [220]:

```
plot_lm_4 = plt.figure();
plt.scatter(modelo_outliers, modelo_norm_resid, alpha=0.5);
sb.regplot(modelo_outliers, modelo_norm_resid,
            scatter=False,
            ci=False,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8});
plot_lm_4.axes[0].set_title('Residuals vs Leverage')
plot_lm_4.axes[0].set_xlabel('Leverage')
plot_lm_4.axes[0].set_ylabel('Standardized Residuals');

# annotations
leverage_top_3 = np.flip(np.argsort(distancia_cook), 0)[:3]
for i in leverage_top_3:
    plot_lm_4.axes[0].annotate(i,
                               xy=(modelo_outliers[i],
                                   modelo_norm_resid[i]))
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation

FutureWarning



Explicação final

MODELO COM ÁRVORE DE DECISÃO

Rodaremos agora a nossa base utilizando o modelo de árvore de decisão e com isso decidiremos qual é o melhor modelo a ser utilizado.

In [221]:

```
Classif_tree = DecisionTreeClassifier(criterion='entropy')
```

In [222]:

```
salarios = pd.read_csv(r'Arquivo_Salarios_Colaboradores_2021.csv', sep=',')
```

Testes da nossa árvore de decisão

Respecto al proceso de selección de accioneros

- qproj_estra
- proj_sustent
- proj_6sigma
- proj_social

In [223]:

```
salarios.head()
```

Out[223]:

	Ordem	salario	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	proj_social	notaavalia
0	1	8000.80	25	4	11	1	1	1	0	79.38
1	2	8500.17	24	5	11	0	0	1	0	84.13
2	3	3350.59	22	1	12	0	0	0	0	46.15
3	4	9500.24	28	4	14	1	0	0	1	83.85
4	5	1500.63	18	2	12	0	0	0	1	73.64

In [224]:

```
X_train, X_test, y_train, y_test = train_test_split(salarios.drop(['qproj_estra', 'Ordem'], axis=1), salarios['qproj_estra'], test_size=0.3, random_state=17)
```

In [225]:

```
classif = Classif tree.fit(X_train, y_train)
```

In [226]:

```
classif.feature importances
```

Out[226]:

```
array([0.3438573, 0.13470208, 0.14362226, 0.11079426, 0.03268949,
       0.03178173, 0.06855475, 0.13399813])
```

In [227]:

```
for feature, importancia in zip(X_train.columns, classif.feature_importances_):
    print("{}:{}".format(feature, (importancia * 100).round()))
```

```
salario:34.0
idade:13.0
tempocasa:14.0
escolar:11.0
proj_sustent:3.0
proj_6sigma:3.0
proj_social:7.0
notaavalia:13.0
```

In [228]:

```
y_pred_all = classif.predict(X_test)
```

In [229]:

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix = confusion_matrix(y_test, y_pred_all)
confusion_matrix
```

Out[229]:

```
array([[ 6,  4,  0,  0,  0,  0,  0,  0],
       [ 3, 20,  3,  1,  0,  0,  0,  0],
       [ 1,  1,  1,  2,  0,  0,  0,  0],
       [ 0,  2,  0,  2,  0,  0,  0,  0],
       [ 1,  3,  0,  0,  0,  0,  0,  0],
```

```
[ 0, 1, 0, 0, 0, 0, 0, 0],
[ 0, 0, 1, 0, 0, 0, 0, 0],
[ 0, 0, 2, 0, 0, 0, 0, 0]])
```

In [230]:

```
print(classification_report(y_test, y_pred_all, zero_division=True))
```

	precision	recall	f1-score	support
0	0.55	0.60	0.57	10
1	0.65	0.74	0.69	27
2	0.14	0.20	0.17	5
3	0.40	0.50	0.44	4
4	1.00	0.00	0.00	4
5	1.00	0.00	0.00	1
6	1.00	0.00	0.00	1
7	1.00	0.00	0.00	2
accuracy			0.54	54
macro avg	0.72	0.26	0.23	54
weighted avg	0.61	0.54	0.50	54

Teste com proj_sustent

In [231]:

```
X_train, X_test, y_train, y_test = train_test_split(salarios.drop(['proj_sustent', 'Ordem'],axis=1),salarios['proj_sustent'],test_size=0.3, random_state=17)
```

In [232]:

```
classif = Classif_tree.fit(X_train, y_train)
```

In [233]:

```
classif.feature_importances_
```

Out[233]:

```
array([0.21229964, 0.15667069, 0.11319798, 0.09998194, 0.2483712 ,
       0.03980373, 0.06312765, 0.06654718])
```

In [234]:

```
for feature,importancia in zip(X_train.columns,classif.feature_importances_):
    print("{}:{}".format(feature, (importancia * 100).round()))
```

```
salario:21.0
idade:16.0
tempocasa:11.0
escolar:10.0
qproj_estra:25.0
proj_6sigma:4.0
proj_social:6.0
notaavalia:7.0
```

In [235]:

```
y_pred_all = classif.predict(X_test)
```

In [236]:

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix = confusion_matrix(y_test, y_pred_all)
confusion_matrix
```

Out[236]:

```
array([[25, 11],
       [10, 20]])
```

[10, 8]])

In [237]:

```
print(classification_report(y_test, y_pred_all, zero_division=True))
```

	precision	recall	f1-score	support
0	0.71	0.69	0.70	36
1	0.42	0.44	0.43	18
accuracy			0.61	54
macro avg	0.57	0.57	0.57	54
weighted avg	0.62	0.61	0.61	54

Teste com proj_6sigma

In [238]:

```
X_train, X_test, y_train, y_test = train_test_split(salarios.drop(['proj_6sigma', 'Ordem'],axis=1),salarios['proj_6sigma'],test_size=0.3, random_state=17)
```

In [239]:

```
classif = Classif_tree.fit(X_train, y_train)
```

In [240]:

```
classif.feature_importances_
```

Out[240]:

```
array([0.27815211, 0.12965474, 0.18307017, 0.1137612 , 0.03176913,
       0.06072852, 0.        , 0.20286414])
```

In [241]:

```
for feature, importancia in zip(X_train.columns,classif.feature_importances_):
    print("{}:{}".format(feature, (importancia * 100).round()))
```

```
salario:28.0
idade:13.0
tempocasa:18.0
escolar:11.0
qproj_estra:3.0
proj_sustent:6.0
proj_social:0.0
notaavalia:20.0
```

In [242]:

```
y_pred_all = classif.predict(X_test)
```

In [243]:

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix = confusion_matrix(y_test, y_pred_all)
confusion_matrix
```

Out[243]:

```
array([[12,  8],
       [ 9, 25]])
```

In [244]:

```
print(classification_report(y_test, y_pred_all, zero_division=True))
```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

	0	0.57	0.60	0.59	20
	1	0.76	0.74	0.75	34
	accuracy		0.69		54
	macro avg	0.66	0.67	0.67	54
	weighted avg	0.69	0.69	0.69	54

Teste com proj_social

In [245]:

```
X_train, X_test, y_train, y_test = train_test_split(salarios.drop(['proj_social', 'Ordem'],axis=1),salarios['proj_social'],test_size=0.3, random_state=17)
```

In [246]:

```
classif = Classif_tree.fit(X_train, y_train)
```

In [247]:

```
classif.feature_importances_
```

Out[247]:

```
array([0.22447858, 0.03189109, 0.16481609, 0.18086928, 0.08681725,
       0.12652239, 0.08757298, 0.09703234])
```

In [248]:

```
for feature,importancia in zip(X_train.columns,classif.feature_importances_):
    print("{}:{}".format(feature, (importancia * 100).round()))
```

```
salario:22.0
idade:3.0
tempocasa:16.0
escolar:18.0
qproj_estra:9.0
proj_sustent:13.0
proj_6sigma:9.0
notaavalia:10.0
```

In [249]:

```
y_pred_all = classif.predict(X_test)
```

In [250]:

```
from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(y_test, y_pred_all)
confusion_matrix
```

Out[250]:

```
array([[18, 4],
       [ 7, 25]])
```

In [251]:

```
print(classification_report(y_test, y_pred_all, zero_division=True))
```

	precision	recall	f1-score	support
0	0.72	0.82	0.77	22
1	0.86	0.78	0.82	32
accuracy			0.80	54
macro avg	0.79	0.80	0.79	54
weighted avg	0.80	0.80	0.80	54

Variaveis para a árvore

In [252]:

```
salarios_dataset = salarios[['proj_social', 'salario', 'idade', 'tempocasa', 'escolar', 'qproj_estra', 'proj_sustent', 'proj_6sigma', 'notaavalia']]
```

Divisão treino-teste

Agora executaremos nosso algoritmo novamente pois conseguimos encontrar a escala que gostaríamos.

In [253]:

```
x = salarios_dataset.values[:, 1:]
y = salarios_dataset.values[:, 0]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)
```

Feature Scaling

In [254]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

Treinamento do modelo

In [255]:

```
def train_model(height):
    model = DecisionTreeClassifier(criterion = 'entropy', max_depth = height, random_state = 0)
    model.fit(x_train, y_train)
    return model
```

Avaliação do modelo

In [256]:

```
for height in range(1, 21): # 1-20
    model = train_model(height)
    y_pred = model.predict(x_test)

    print('-----\n')
    print(f'Altura - {height}\n')
    print("Precisão: " + str(accuracy_score(y_test, y_pred)))
```

```
-----
Altura - 1
Precisão: 0.75
-----
Altura - 2
Precisão: 0.8055555555555556
-----
Altura - 3
Precisão: 0.6944444444444444
-----
Altura - 4
Precisão: 0.7222222222222222
-----
```

Altura - 5

Precisão: 0.6944444444444444

Altura - 6

Precisão: 0.75

Altura - 7

Precisão: 0.7222222222222222

Altura - 8

Precisão: 0.75

Altura - 9

Precisão: 0.75

Altura - 10

Precisão: 0.75

Altura - 11

Precisão: 0.75

Altura - 12

Precisão: 0.75

Altura - 13

Precisão: 0.75

Altura - 14

Precisão: 0.75

Altura - 15

Precisão: 0.75

Altura - 16

Precisão: 0.75

Altura - 17

Precisão: 0.75

Altura - 18

Precisão: 0.75

Altura - 19

Precisão: 0.75

Altura - 20

Precisão: 0.75

```
from IPython.display import Image
from sklearn.tree import export_graphviz
```

```
model = train_model(3)
```

```
feature_names = ['salario', 'idade', 'tempocasa', 'escolar', 'qproj_estra', 'proj_sustent', 'proj_6sigma', 'notaavalua']
```

```
classes_names = ['%.f' % i for i in model.classes_]
```

```
dot_data = export_graphviz(model, filled=True, feature_names=feature_names, class_names=classes_names, rounded=True, special_characters=True)
```

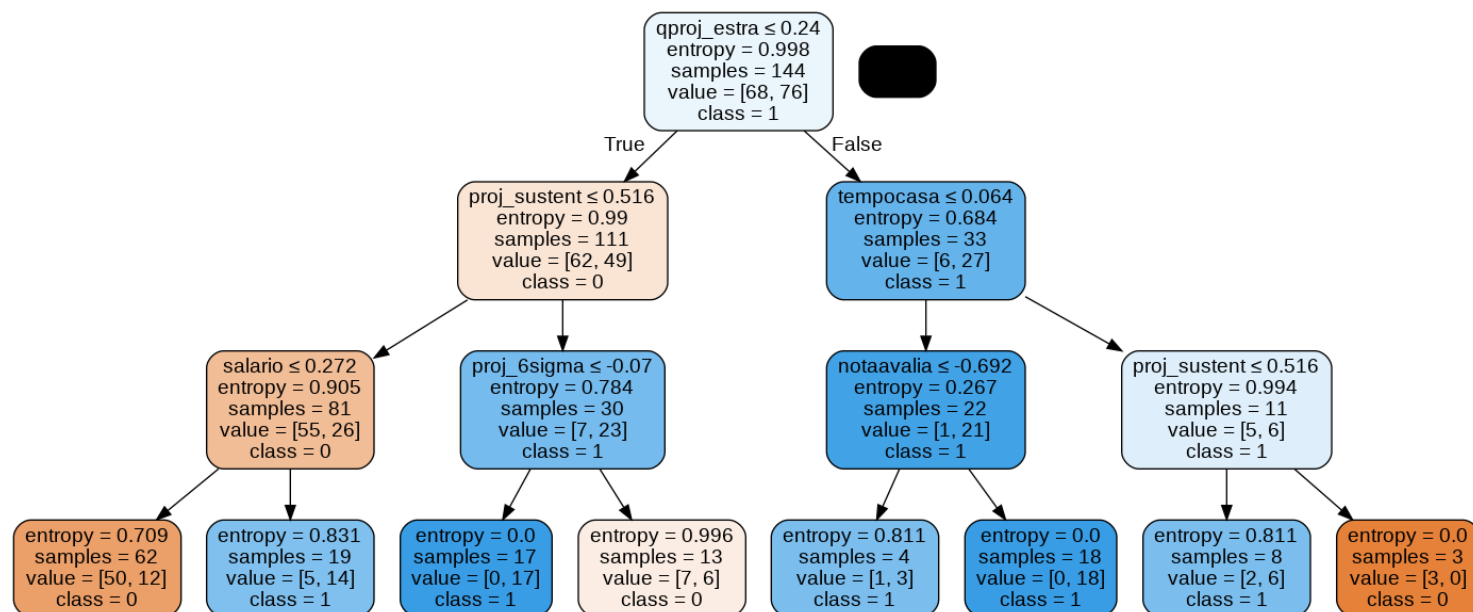
```
graph = pydotplus.graph_from_dot_data(dot_data)
```

```
Image(graph.create_png())
```

```
graph.write_png("tree.png")
```

```
Image("tree.png")
```

Out[257]:



Resultado final da árvore

In [258]:

```
X_train, X_test, y_train, y_test = train_test_split(salarios.drop(['proj_social', 'Ordem'], axis=1), salarios['proj_social'], test_size=0.3, random_state=17)
```

```
X_train.shape, X_test.shape
```

Out[258]:

```
((126, 8), (54, 8))
```

In [259]:

```
classif = Classif_tree.fit(X_train, y_train)
```

In [260]:

```
classif.feature_importances_
```

Out[260]:

```
array([0.22655583, 0.05275821, 0.21878805, 0.1298255 , 0.08681725,
       0.12652239, 0.08757298, 0.0711598 ])
```

In [261]:

```
for feature, importancia in zip(X_train.columns, classif.feature_importances_):
    print("{}:{}".format(feature, importancia))
```

```
salario:0.22655582736793717
```

```
idade:0.052758205004239674
```

```
tempocasa:0.21878804853045838
```

tempocasa:0.21078004000040000
escolar:0.12982550239043558
qproj_estra:0.08681724612937822
proj_sustent:0.12652239196619702
proj_6sigma:0.0875729764712559
notaavalia:0.07115980214009798

In [262]:

```
y_pred_all = classif.predict(X_test)
```

In [263]:

```
from sklearn.metrics import confusion_matrix
# Matriz de Confusão

confusion_matrix = confusion_matrix(y_test, y_pred_all)
confusion_matrix
```

Out[263]:

```
array([[15,  7],
       [ 7, 25]])
```

In [264]:

```
print(classification_report(y_test, y_pred_all))
```

	precision	recall	f1-score	support
0	0.68	0.68	0.68	22
1	0.78	0.78	0.78	32
accuracy			0.74	54
macro avg	0.73	0.73	0.73	54
weighted avg	0.74	0.74	0.74	54

In [265]:

```
salarios.drop(['proj_social', 'Ordem'],axis=1)
```

Out[265]:

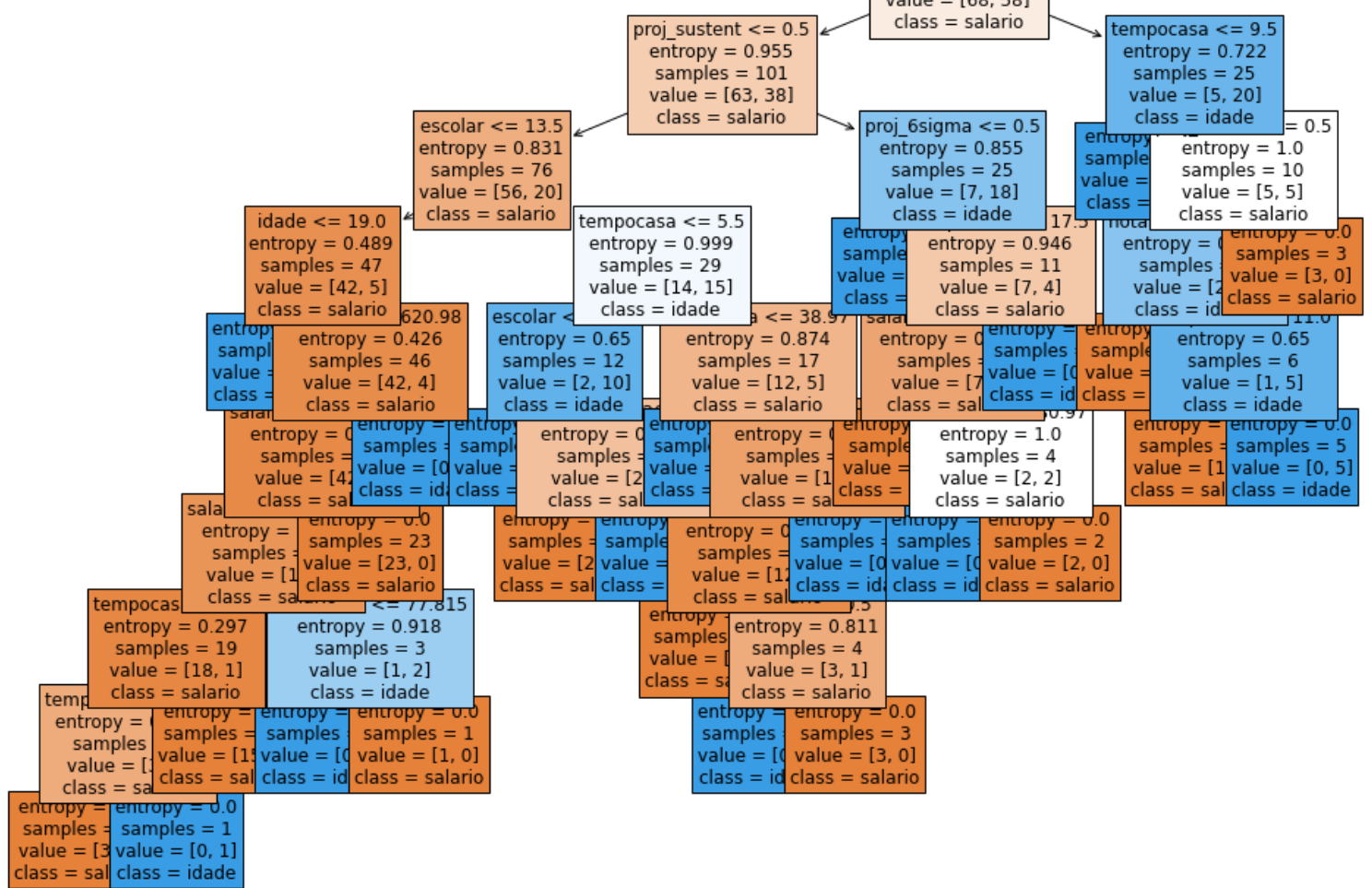
	salario	idade	tempocasa	escolar	qproj_estra	proj_sustent	proj_6sigma	notaavalia
0	8000.80	25	4	11	1	1	1	79.38
1	8500.17	24	5	11	0	0	1	84.13
2	3350.59	22	1	12	0	0	0	46.15
3	9500.24	28	4	14	1	0	0	83.85
4	1500.63	18	2	12	0	0	0	73.64
...
175	7896.70	36	14	13	0	1	1	88.87
176	10575.13	26	15	14	1	0	1	85.45
177	6309.66	36	3	14	1	0	0	76.72
178	2100.68	28	11	12	0	1	0	76.22
179	4059.13	21	3	15	1	1	0	79.40

180 rows x 8 columns

In [266]:

```
fig = plt.figure(figsize=(16,12))
a = plot_tree(Classif_tree, feature_names=X_train.columns, fontsize=12,
filled=True,
class_names=['salario', 'idade', 'tempocasa', 'escolar', 'qproj_estra', 'proj_sustent', 'proj_6sigma', 'notaavalia'])
```

qproj_estra <= 1.5
entropy = 0.995
samples = 126
value = [68, 58]



Resultado da análise

Nós verificamos que para a variável alvo que é salário o melhor modelo é o da regressão linear, pois com ela conseguimos perceber melhor a relação que existe entre o funcionário da empresa obter um melhor salário a partir dos projetos entregues, idade, escolaridade e tempo de casa. O modelo nós mostrou isso e facilitou nossa análise. Todavia, percebemos também que na árvore chegamos na conclusão de que os projetos interferem na questão do salário, porém com a árvore achamos que ficou menos performático para chegar nesta conclusão.