Carolina_Rangel_Lista

May 15, 2022

[1]:

```python
# Core
import pandas as pd
import numpy as np
from scipy.stats import skew
from google.colab import files
import io

# Visual
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns

# Models
from sklearn.linear_model import LinearRegression, Ridge, RidgeCV, ElasticNet,␣
↪LassoCV, ElasticNetCV
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from scipy import stats as st
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,␣
↪AdaBoostRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import VarianceThreshold
from scipy.stats import chi2_contingency

from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

1 EXERCÍCIO 1

[2]:
```
uploaded = files.upload()
df = pd.read_csv(io.BytesIO(uploaded['house-prices.csv']))
```

```
<IPython.core.display.HTML object>
```

```
Saving house-prices.csv to house-prices (1).csv
```

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
```

```
31  BsmtCond       1423 non-null   object
32  BsmtExposure   1422 non-null   object
33  BsmtFinType1   1423 non-null   object
34  BsmtFinSF1     1460 non-null   int64
35  BsmtFinType2   1422 non-null   object
36  BsmtFinSF2     1460 non-null   int64
37  BsmtUnfSF      1460 non-null   int64
38  TotalBsmtSF    1460 non-null   int64
39  Heating        1460 non-null   object
40  HeatingQC      1460 non-null   object
41  CentralAir     1460 non-null   object
42  Electrical     1459 non-null   object
43  1stFlrSF       1460 non-null   int64
44  2ndFlrSF       1460 non-null   int64
45  LowQualFinSF   1460 non-null   int64
46  GrLivArea      1460 non-null   int64
47  BsmtFullBath   1460 non-null   int64
48  BsmtHalfBath   1460 non-null   int64
49  FullBath       1460 non-null   int64
50  HalfBath       1460 non-null   int64
51  BedroomAbvGr   1460 non-null   int64
52  KitchenAbvGr   1460 non-null   int64
53  KitchenQual    1460 non-null   object
54  TotRmsAbvGrd   1460 non-null   int64
55  Functional     1460 non-null   object
56  Fireplaces     1460 non-null   int64
57  FireplaceQu    770 non-null    object
58  GarageType     1379 non-null   object
59  GarageYrBlt    1379 non-null   float64
60  GarageFinish   1379 non-null   object
61  GarageCars     1460 non-null   int64
62  GarageArea     1460 non-null   int64
63  GarageQual     1379 non-null   object
64  GarageCond     1379 non-null   object
65  PavedDrive     1460 non-null   object
66  WoodDeckSF     1460 non-null   int64
67  OpenPorchSF    1460 non-null   int64
68  EnclosedPorch  1460 non-null   int64
69  3SsnPorch      1460 non-null   int64
70  ScreenPorch    1460 non-null   int64
71  PoolArea       1460 non-null   int64
72  PoolQC         7 non-null      object
73  Fence          281 non-null    object
74  MiscFeature    54 non-null     object
75  MiscVal        1460 non-null   int64
76  MoSold         1460 non-null   int64
77  YrSold         1460 non-null   int64
78  SaleType       1460 non-null   object
```

```
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

[4]:
```python
# Colocando a variável dependente como primeira no Dataset

cols_to_order = ['SalePrice']
new_columns = cols_to_order + (df.columns.drop(cols_to_order).tolist())
df = df[new_columns]
df.head()
```

[4]:
```
   SalePrice  Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley  \
0     208500   1          60       RL         65.0     8450   Pave   NaN
1     181500   2          20       RL         80.0     9600   Pave   NaN
2     223500   3          60       RL         68.0    11250   Pave   NaN
3     140000   4          70       RL         60.0     9550   Pave   NaN
4     250000   5          60       RL         84.0    14260   Pave   NaN

  LotShape LandContour  … ScreenPorch PoolArea PoolQC Fence MiscFeature  \
0      Reg         Lvl  …           0        0    NaN   NaN         NaN
1      Reg         Lvl  …           0        0    NaN   NaN         NaN
2      IR1         Lvl  …           0        0    NaN   NaN         NaN
3      IR1         Lvl  …           0        0    NaN   NaN         NaN
4      IR1         Lvl  …           0        0    NaN   NaN         NaN

  MiscVal MoSold YrSold  SaleType  SaleCondition
0       0      2   2008        WD         Normal
1       0      5   2007        WD         Normal
2       0      9   2008        WD         Normal
3       0      2   2006        WD        Abnorml
4       0     12   2008        WD         Normal

[5 rows x 81 columns]
```

2 Parte I: Tratamento dos Dados

2.1 Missing values geral

[5]:
```python
total = df.isnull().sum().sort_values(ascending=False)

percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)

missing_data = pd.concat(
    [total, percent],
```

```python
      axis=1,
      keys=['Total', 'Percent'])

missing_data.head(20)
```

[5]:

```
              Total   Percent
PoolQC         1453  0.995205
MiscFeature    1406  0.963014
Alley          1369  0.937671
Fence          1179  0.807534
FireplaceQu     690  0.472603
LotFrontage     259  0.177397
GarageFinish     81  0.055479
GarageType       81  0.055479
GarageYrBlt      81  0.055479
GarageQual       81  0.055479
GarageCond       81  0.055479
BsmtExposure     38  0.026027
BsmtFinType2     38  0.026027
BsmtFinType1     37  0.025342
BsmtCond         37  0.025342
BsmtQual         37  0.025342
MasVnrType        8  0.005479
MasVnrArea        8  0.005479
Electrical        1  0.000685
KitchenAbvGr      0  0.000000
```

Há algumas variáveis que tem mais de 45% de missing values. A falta destes valores pode comprometer o modelo e sua acurácia e, por isso, irei removê-las.

[6]:
```python
lista = missing_data[missing_data['Percent'] > 0.45].index
```

[7]:
```python
df = df.drop(lista, 1)
```

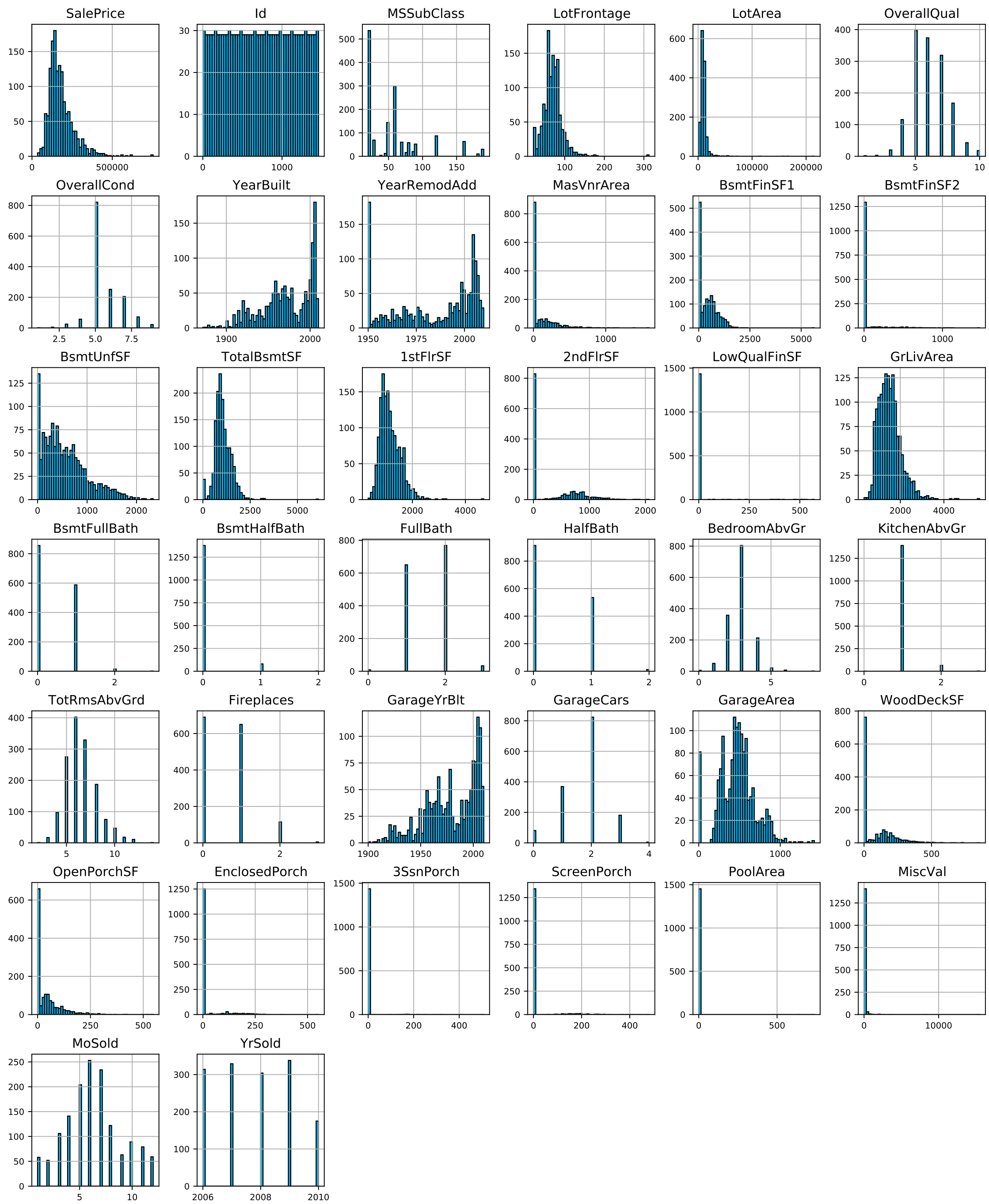
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

Lidarei com o restante dos missing values dentro da sua categoria de numerico, categórico ou binário.

2.2 Variáveis numéricas

[8]:
```
numerical_cols = [cname for cname in df.columns if
                  df[cname].dtype in ['int64', 'float64']]

df_num = df[numerical_cols].copy()
```

[9]:
```
# Dando uma olhada na distribuição destas variáveis

fig_ = df_num.hist(figsize=(16, 20), bins=50, color="deepskyblue",
                           edgecolor="black", xlabelsize=8, ylabelsize=8)
```



Olhando as distribuições, é possível perceber que há variáveis que variam pouco em seu valor. Pouca variação nos dados leva a pouco efeito dentro de um modelo e, por isso, irei remover todas as variáveis onde 80% dos dados sejam constantes.

[10]:
```
sel = VarianceThreshold(threshold=0.2)
sel.fit(df_num.iloc[:, :-1])
```

```
print(f"Variáveis mantidas: {sum(sel.get_support())}")
print(f"\nVariáveis quase constantes: {len(df_num.iloc[:, :-1].columns) -␣
↪sum(sel.get_support())}")

quasi_constant_features_list = [x for x in df_num.iloc[:, :-1].columns if x not␣
↪in df_num.iloc[:, :-1].columns[sel.get_support()]]

print(f"\nVariáveis para dropar: {quasi_constant_features_list}")
```

```
Variáveis mantidas: 35

Variáveis quase constantes: 2

Variáveis para dropar: ['BsmtHalfBath', 'KitchenAbvGr']
```

[11]:
```
df_num.drop(quasi_constant_features_list, axis=1, inplace=True)
```

[12]:
```
total = df_num.isnull().sum().sort_values(ascending=False)

percent = (df_num.isnull().sum()/df_num.isnull().count()).
↪sort_values(ascending=False)

missing_data = pd.concat(
     [total, percent],
      axis=1,
      keys=['Total', 'Percent'])
missing_data.head(5)
```

[12]:

	Total	Percent
LotFrontage	259	0.177397
GarageYrBlt	81	0.055479
MasVnrArea	8	0.005479
OpenPorchSF	0	0.000000
BedroomAbvGr	0	0.000000

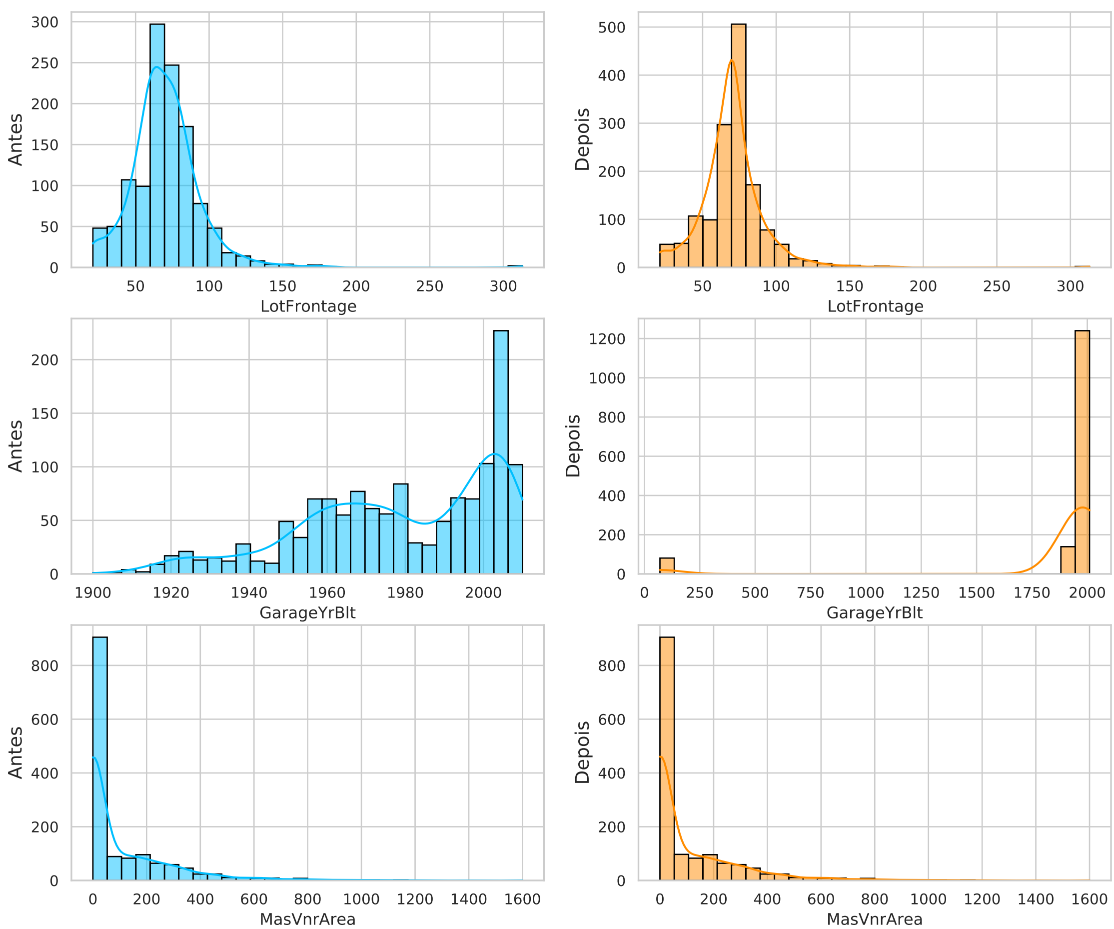
Irei fazer o input destas três variáveis da forma mais simples, pela média da coluna.

[13]:
```
df_num_input = df_num.copy()
```

[14]:
```
df_num_input = df_num_input.fillna(df_num['LotFrontage'].mean())
df_num_input = df_num_input.fillna(df_num['GarageYrBlt'].mean())
df_num_input = df_num_input.fillna(df_num['MasVnrArea'].mean())
```

[15]:
```
# Agora vamos olhar como as variáveis mudaram antes e depois do input dos dados

sns.set(rc={"figure.figsize": (14, 12)})
```

```
sns.set_style("whitegrid")
fig, axes = plt.subplots(3, 2)


for feature, fig_pos in zip(["LotFrontage", "GarageYrBlt", "MasVnrArea"], [0,␣
↪1, 2]):

    """Distribuição antes e depois de input"""

    # antes
    p = sns.histplot(ax=axes[fig_pos, 0], x=df_num[feature],
                     kde=True, bins=30, color="deepskyblue", edgecolor="black")
    p.set_ylabel(f"Antes", fontsize=14)

    # depois
    q = sns.histplot(ax=axes[fig_pos, 1], x=df_num_input[feature],
                     kde=True, bins=30, color="darkorange", edgecolor="black")
    q.set_ylabel(f"Depois", fontsize=14)
```



Olhando as mudanças, irei remover GarageYrBlt e manter as outras duas

[16]:
```
df_num = df_num.fillna(df_num['LotFrontage'].mean())
df_num = df_num.fillna(df_num['MasVnrArea'].mean())
```

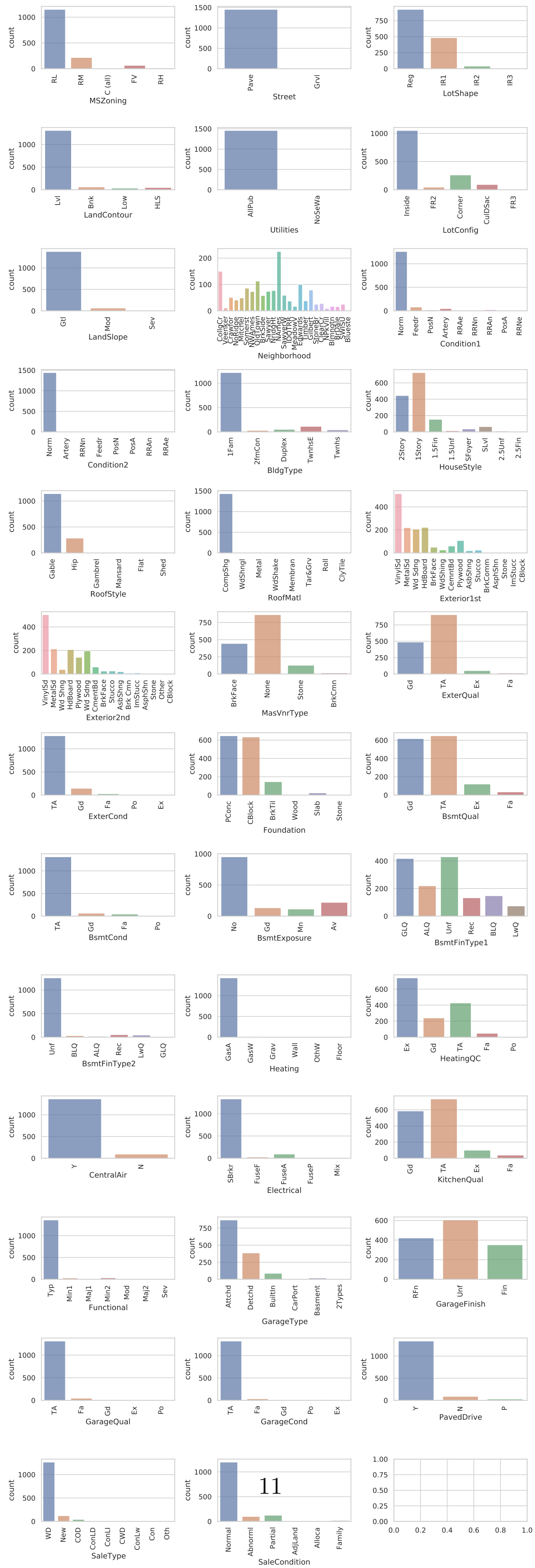
[17]:
```
df_num.isnull().sum().max() # não temos mais missing values nas variáveis␣
↪numéricas
```

[17]: 0

2.3 Variáveis categóricas

[18]:
```
df_cat = [
    i for i in df.columns if df.dtypes[i] == "object"]
df_cat.append("SalePrice")

df_cat = df[df_cat]
```

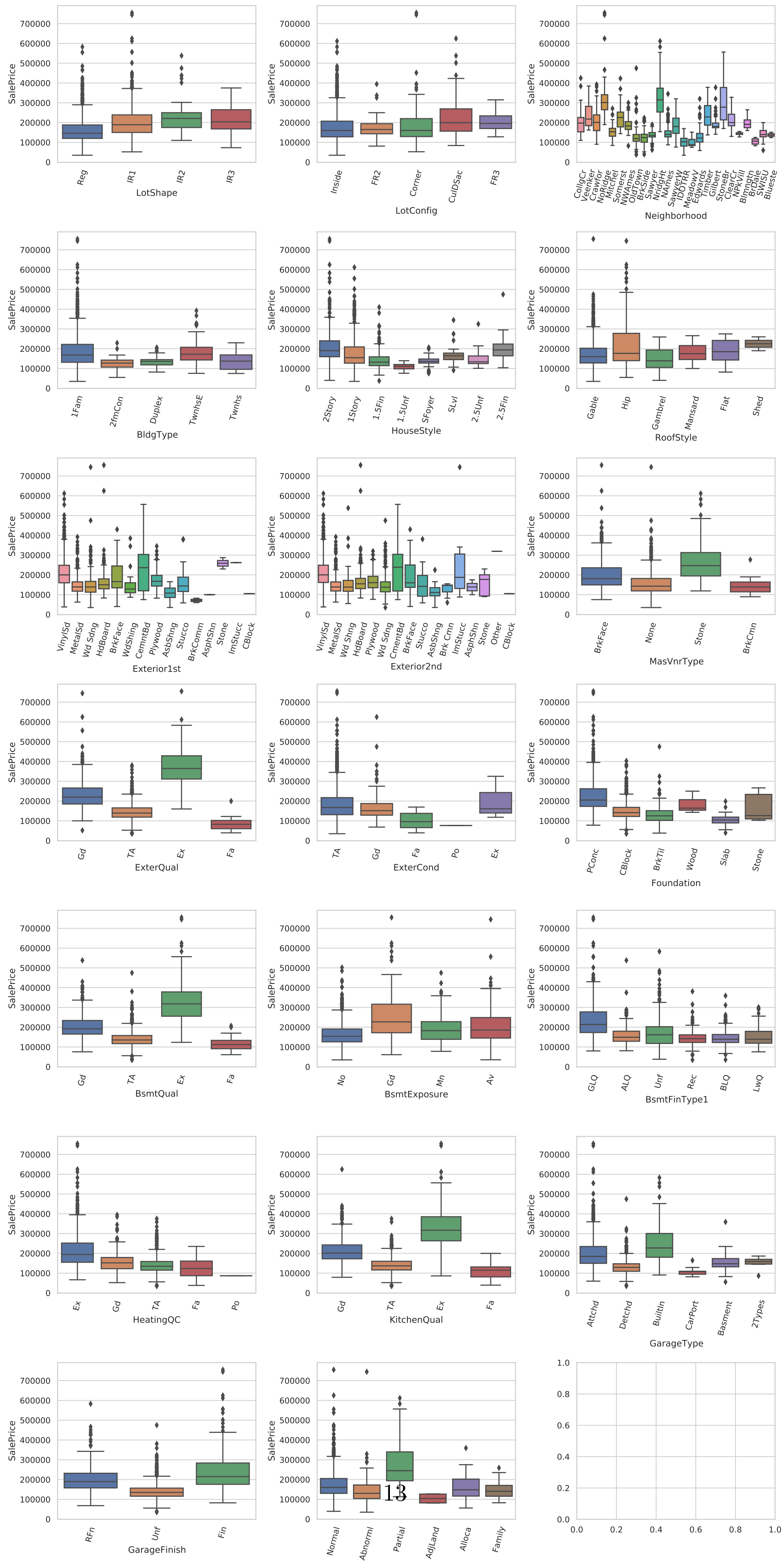
[19]:
```
# Assim como nas variáveis numéricas, irei olhar a distribuição para saber se␣
↪há variáveis muito constantes que podemos dropar por não terem efeito

fig, axes = plt.subplots(round(len(df_cat.columns) / 3), 3, figsize=(12, 35))

for i, ax in enumerate(fig.axes):
    # plot barplot of each feature
    if i < len(df_cat.columns) - 1:
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=90)
        sns.countplot(x=df_cat.columns[i], alpha=0.7, data=df_cat, ax=ax)

fig.tight_layout()
```


count
MSZoning
RL RM C (all) FV RH
Street
Pave Grvl
LotShape
Reg IR1 IR2 IR3
LandContour
Lvl Bnk Low HLS
Utilities
AllPub NoSeWa
LotConfig
Inside FR2 Corner CulDSac FR3
LandSlope
Gtl Mod Sev
Neighborhood
Condition1
Norm Feedr PosN Artery RRAe RRNn RRAn PosA RRNe
Condition2
Norm Artery RRNn Feedr PosN PosA RRAn RRAe
BldgType
1Fam 2fmCon Duplex TwnhsE Twnhs
HouseStyle
2Story 1Story 1.5Fin 1.5Unf SFoyer SLvl 2.5Unf 2.5Fin
RoofStyle
Gable Hip Gambrel Mansard Flat Shed
RoofMatl
CompShg WdShngl Metal WdShake Membran Tar&Grv Roll ClyTile
Exterior1st
VinylSd MetalSd Wd Sdng HdBoard BrkFace WdShing CemntBd Plywood AsbShng Stucco BrkComm AsphShn Stone ImStucc CBlock
Exterior2nd
VinylSd MetalSd Wd Shng HdBoard Plywood Wd Sdng CmentBd BrkFace Stucco AsbShng Brk Cmn ImStucc AsphShn Stone Other CBlock
MasVnrType
BrkFace None Stone BrkCmn
ExterQual
Gd TA Ex Fa
ExterCond
TA Gd Fa Po Ex
Foundation
PConc CBlock BrkTil Wood Slab Stone
BsmtQual
Gd TA Ex Fa
BsmtCond
TA Gd Fa Po
BsmtExposure
No Gd Mn Av
BsmtFinType1
GLQ ALQ Unf Rec BLQ LwQ
BsmtFinType2
Unf BLQ ALQ Rec LwQ GLQ
Heating
GasA GasW Grav Wall OthW Floor
HeatingQC
Ex Gd TA Fa Po
CentralAir
Y N
Electrical
SBrkr FuseF FuseA FuseP Mix
KitchenQual
Gd TA Ex Fa
Functional
Typ Min1 Maj1 Min2 Mod Maj2 Sev
GarageType
Attchd Detchd BuiltIn CarPort Basment 2Types
GarageFinish
RFn Unf Fin
GarageQual
TA Fa Gd Ex Po
GarageCond
TA Fa Gd Po Ex
PavedDrive
Y N P
SaleType
WD New COD ConLD ConLI CWD ConLw Con Oth
SaleCondition
Normal Abnorml Partial AdjLand Alloca Family

[20]:
```
# Há variáveis claramente dominadas por apenas um valor. Como variáveis muito␣
↪constantes não tem efeito no modelo, irei removê-las. São elas:

cols_to_drop = [
     'MSZoning',
    'Street',
    'LandContour',
    'Utilities',
    'LandSlope',
    'Condition1',
    'Condition2',
    'RoofMatl',
    'BsmtCond',
    'BsmtFinType2',
    'Heating',
    'CentralAir',
    'Electrical',
    'Functional',
    'GarageQual',
    'GarageCond',
    'PavedDrive',
    'SaleType'
]

df_cat = df_cat.drop(cols_to_drop, 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:24: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
```

[21]:
```
# Agora, por meio de box plots, podemos olhar a distribuição de cada variável␣
↪em relação a nossa variável dependente

fig, axes = plt.subplots(
    round(len(df_cat.columns)/3), 3, figsize=(15, 30))

for i, ax in enumerate(fig.axes):

    if i < len(df_cat.columns) - 1:
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=75)
        sns.boxplot(
            x=df_cat.columns[i], y="SalePrice", data=df_cat, ax=ax)

fig.tight_layout()
```


SalePrice
LotShape
Reg IR1 IR2 IR3
LotConfig
Inside FR2 Corner CulDSac FR3
Neighborhood
BldgType
1Fam 2fmCon Duplex TwnhsE Twnhs
HouseStyle
2Story 1Story 1.5Fin 1.5Unf SFoyer SLvl 2.5Unf 2.5Fin
RoofStyle
Gable Hip Gambrel Mansard Flat Shed
Exterior1st
Exterior2nd
MasVnrType
BrkFace None Stone BrkCmn
ExterQual
Gd TA Ex Fa
ExterCond
TA Gd Fa Po Ex
Foundation
PConc CBlock BrkTil Wood Slab Stone
BsmtQual
Gd TA Ex Fa
BsmtExposure
No Gd Mn Av
BsmtFinType1
GLQ ALQ Unf Rec BLQ LwQ
HeatingQC
Ex Gd TA Fa Po
KitchenQual
Gd TA Ex Fa
GarageType
Attchd Detchd BuiltIn CarPort Basment 2Types
GarageFinish
RFn Unf Fin
SaleCondition
Normal Abnorml Partial AdjLand Alloca Family

É notável que algumas variáveis possuem distribuições muito parecidas de SalePrice. Por isso, é interessante olharmos se elas são muito co-dependentes e, caso sim, podemos remover uma do dataset.

Especificamente, estes pares se paracem:

- "Exterior1st" e "Exterior2nd"
- "ExterQual" e "MasVnrType"
- "BsmtQual" e "BsmtExposure"

[22]:
```python
# O código para realizar esse tipo de teste foi pego do kaggle https://www.
↪kaggle.com/code/harvindarjunrai

sns.set(rc={"figure.figsize": (10, 7)})

X = ["Exterior1st", "ExterQual", "BsmtQual"]
Y = ["Exterior2nd", "MasVnrType", "BsmtExposure"]

# Parameters for Chi-squared test (5% significance level)
prob = 0.95
alpha = 1.0 - prob

for i, j in zip(X, Y):

    # Contingency table
    cont = df_cat[[i, j]].pivot_table(
        index=i, columns=j, aggfunc=len, margins=True, margins_name="Total")
    tx = cont.loc[:, ["Total"]]
    ty = cont.loc[["Total"], :]
    n = len(df_cat)
    indep = tx.dot(ty) / n
    c = cont.fillna(0)  # Replace NaN with 0 in the contingency table
    measure = (c - indep) ** 2 / indep
    xi_n = measure.sum().sum()
    table = measure / xi_n

    # Performing Chi-sq test
    CrosstabResult = pd.crosstab(
        index=df_cat[i], columns=df_cat[j])
    ChiSqResult = chi2_contingency(CrosstabResult)
    # P-Value is the Probability of H0 being True
    print(f"P-Value of the ChiSq Test bewteen {i} and {j} is:␣
↪{ChiSqResult[1]}\n")
    print('significance=%.3f, p=%.3f' % (alpha, ChiSqResult[1]))
    if ChiSqResult[1] <= alpha:
        print('Dependent (reject H0)')
```

```
    else:
        print('Independent (fail to reject H0)')
```

```
P-Value of the ChiSq Test bewteen Exterior1st and Exterior2nd is: 0.0

significance=0.050, p=0.000
Dependent (reject H0)
P-Value of the ChiSq Test bewteen ExterQual and MasVnrType is:
1.0187554679218715e-54

significance=0.050, p=0.000
Dependent (reject H0)
P-Value of the ChiSq Test bewteen BsmtQual and BsmtExposure is:
3.879215036512606e-32

significance=0.050, p=0.000
Dependent (reject H0)
```

Como há co-dependência destas variáveis, uma de cada par pode ser dropada

[23]:
```
df_cat = df_cat.drop(Y, 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

[24]:
```
# Agora vamos olhar os missing values

total = df_cat.isnull().sum().sort_values(ascending=False)

percent = (df_cat.isnull().sum()/df_cat.isnull().count()).
 ↪sort_values(ascending=False)

missing_data = pd.concat(
     [total, percent],
      axis=1,
      keys=['Total', 'Percent'])

missing_data.head(5)
```

[24]:

	Total	Percent
GarageFinish	81	0.055479
GarageType	81	0.055479
BsmtFinType1	37	0.025342
BsmtQual	37	0.025342
LotShape	0	0.000000

[25]:
```
# Esse método de input também foi pego do kaggle mencionado acima

categ_fill_null = {"GarageType": df_cat["GarageType"].mode().iloc[0],
                   "GarageFinish": df_cat["GarageFinish"].mode().iloc[0],
                   "BsmtQual": df_cat["BsmtQual"].mode().iloc[0],
                   "BsmtFinType1": df_cat["BsmtFinType1"].mode().iloc[0]}

df_cat = df_cat.fillna(value=categ_fill_null)
```

[26]:
```
df_cat.isnull().sum().max() # Não temos mais missing values nas variáveis␣
 ↪categóricas
```

[26]: 0

Agora precisamos transformar as variáveis categóricas em dummies para poder colocá-las no modelo

[27]:
```
df_cat = df_cat.drop(["SalePrice"], 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

[28]:
```
df_dum = pd.get_dummies(df_cat, drop_first=True)
```

[29]:
```
df_dum
```

[29]:
```
      LotShape_IR2  LotShape_IR3  LotShape_Reg  LotConfig_CulDSac  \
0                0             0             1                  0
1                0             0             1                  0
2                0             0             0                  0
3                0             0             0                  0
4                0             0             0                  0
…              …             …             …                  …
1455             0             0             1                  0
1456             0             0             1                  0
1457             0             0             1                  0
1458             0             0             1                  0
1459             0             0             1                  0

      LotConfig_FR2  LotConfig_FR3  LotConfig_Inside  Neighborhood_Blueste  \
0                 0              0                 1                     0
1                 1              0                 0                     0
2                 0              0                 1                     0
3                 0              0                 0                     0
4                 1              0                 0                     0
…               …              …                 …                     …
```

```
1455              0              0                1                     0
1456              0              0                1                     0
1457              0              0                1                     0
1458              0              0                1                     0
1459              0              0                1                     0

      Neighborhood_BrDale  Neighborhood_BrkSide  …  GarageType_BuiltIn  \
0                       0                     0  …                   0
1                       0                     0  …                   0
2                       0                     0  …                   0
3                       0                     0  …                   0
4                       0                     0  …                   0
…                     …                     …  …                 …
1455                    0                     0  …                   0
1456                    0                     0  …                   0
1457                    0                     0  …                   0
1458                    0                     0  …                   0
1459                    0                     0  …                   0

      GarageType_CarPort  GarageType_Detchd  GarageFinish_RFn  \
0                      0                  0                 1
1                      0                  0                 1
2                      0                  0                 1
3                      0                  1                 0
4                      0                  0                 1
…                    …                  …                 …
1455                   0                  0                 1
1456                   0                  0                 0
1457                   0                  0                 1
1458                   0                  0                 0
1459                   0                  0                 0

      GarageFinish_Unf  SaleCondition_AdjLand  SaleCondition_Alloca  \
0                    0                      0                     0
1                    0                      0                     0
2                    0                      0                     0
3                    1                      0                     0
4                    0                      0                     0
…                  …                      …                     …
1455                 0                      0                     0
1456                 1                      0                     0
1457                 0                      0                     0
1458                 1                      0                     0
1459                 0                      0                     0

      SaleCondition_Family  SaleCondition_Normal  SaleCondition_Partial
0                        0                     1                      0
```

```
1                       0                      1                      0
2                       0                      1                      0
3                       0                      0                      0
4                       0                      1                      0
…                   …                    …                    …
1455                    0                      1                      0
1456                    0                      1                      0
1457                    0                      1                      0
1458                    0                      1                      0
1459                    0                      1                      0

[1460 rows x 100 columns]
```

2.4 Criando variáveis

Há duas variáveis que trazem informações sobre o ano que a casa foi construída e o ano que foi reformada. Entretanto, parece ser mais informativo termos a informação de quanto tempo faz que essas coisas aconteceram do que o ano que aconteceram. Então, ireia fazer essa tranformação

[30]:
```python
# Primeiro precisamos juntar as variáveis dummies e as numéricas de volta

df_n = pd.concat([df_num, df_dum], axis = 1)
print(f"DF: {df_n.shape}")
```

```
DF: (1460, 136)
```

[31]:
```python
df_n["AgeSinceConst"] = (df_n["YearBuilt"].max() - df_n["YearBuilt"])
df_n = df_n.drop(["YearBuilt"], 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
```

[32]:
```python
df_n["AgeSinceRemod"] = (df_n["YearRemodAdd"].max() - df_n["YearRemodAdd"])
df_n = df_n.drop(["YearRemodAdd"], 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
```

2.5 Train test split nos dados

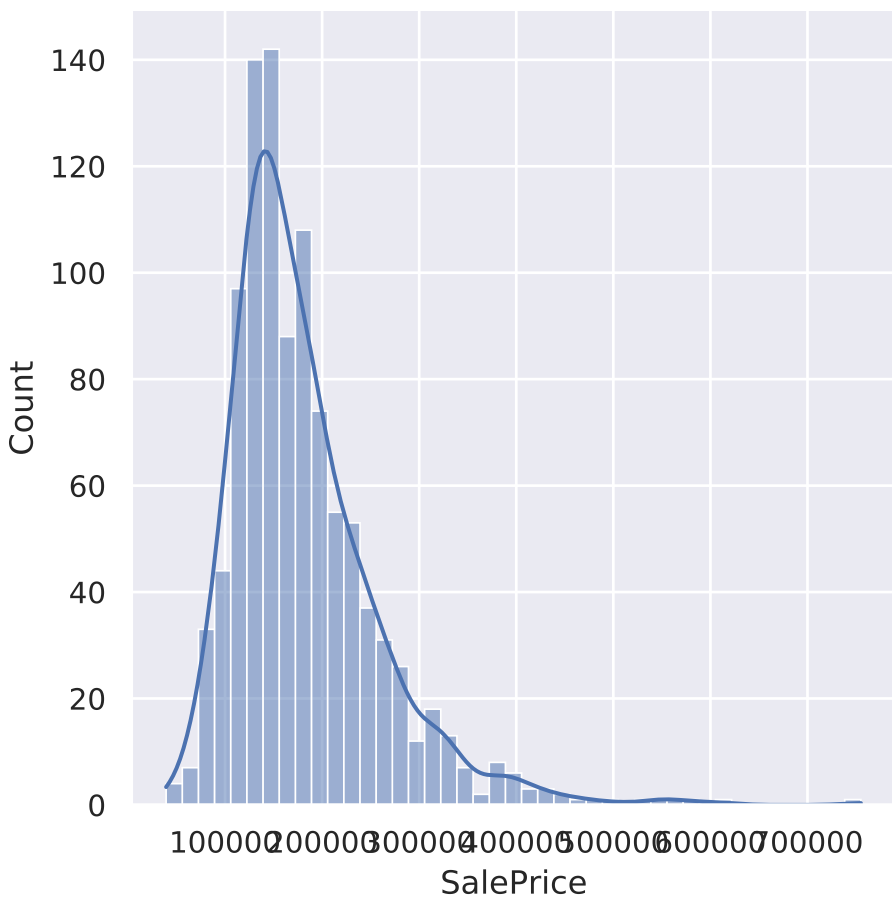
[33]:
```
df_n = df_n.drop('Id', 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

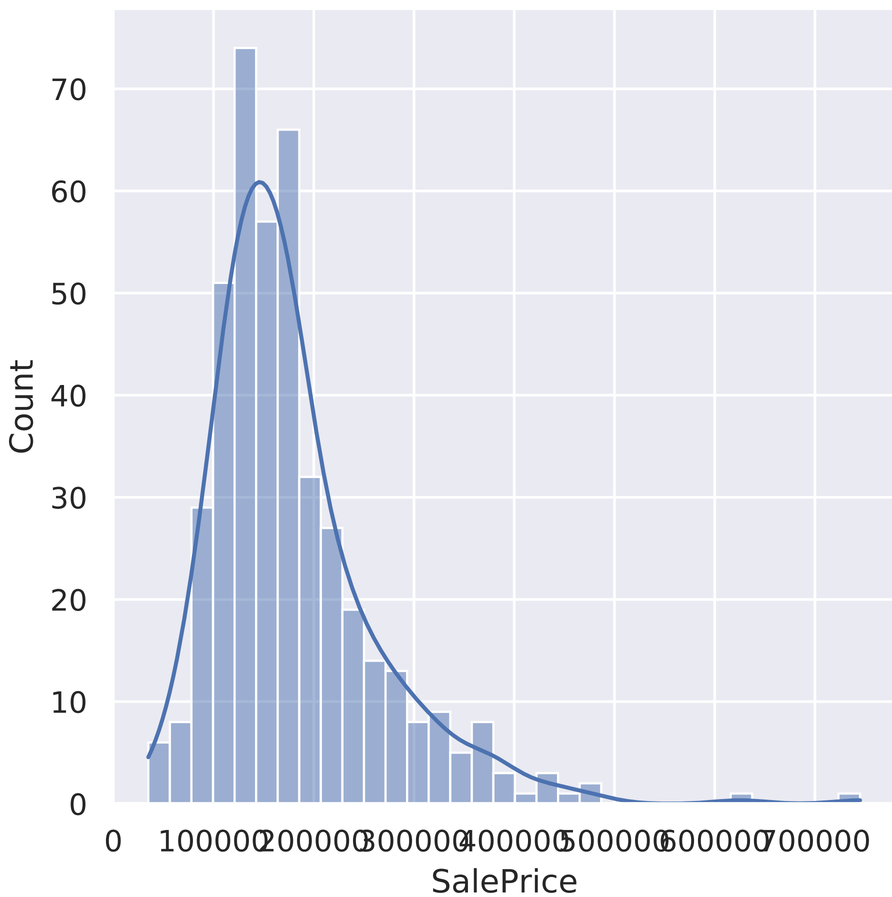
[34]:
```
train, test = train_test_split(df_n, test_size=0.3, random_state = 1)
```

2.6 Olhando a variável dependente

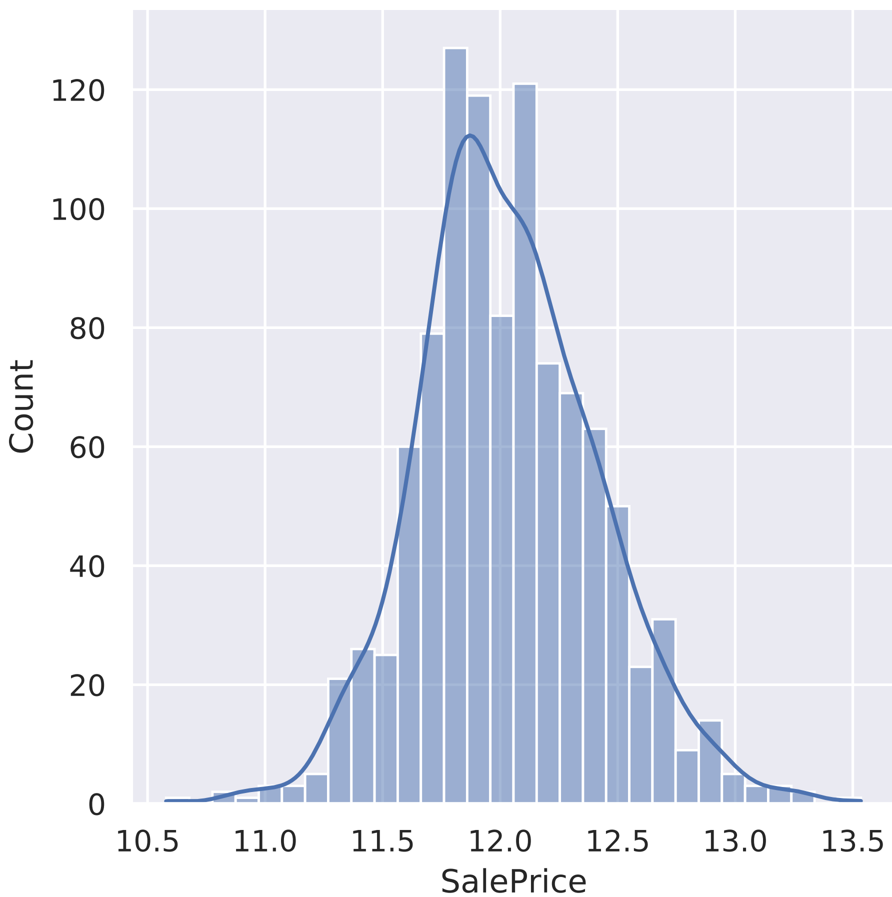
[35]:
```
sns.displot(train['SalePrice'], kde=True);
```



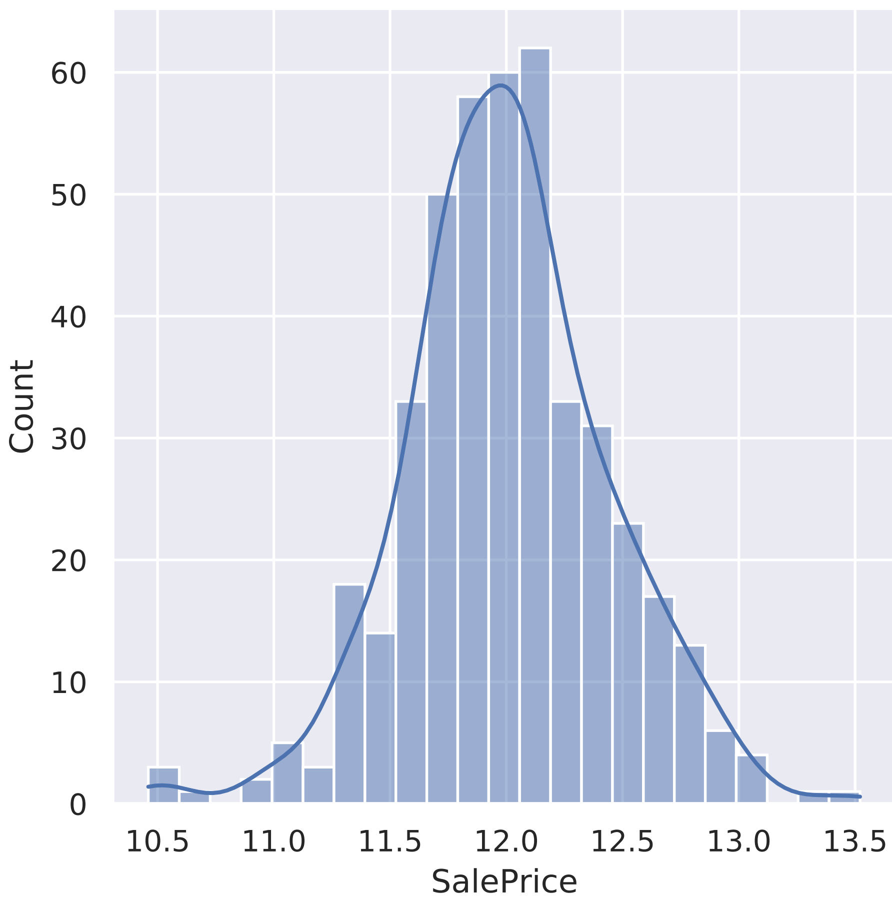
[36]:
```
sns.displot(test['SalePrice'], kde=True);
```



[37]:
```
sns.displot(np.log1p(train['SalePrice']), kde=True);
```



[38]:
```
sns.displot(np.log1p(test['SalePrice']), kde=True);
```



É possível perceber que a variável dependente tende para a esquerda. Para tornar sua distribuição normal, irei usar o log.

[39]:
```
train['SalePrice'] = np.log1p(train['SalePrice'])
test['SalePrice'] = np.log1p(test['SalePrice'])
```

2.7 Correlation Matrix

[40]:
```
corr_matrix = train.corr().sort_values(
    by = "SalePrice",  ascending=False, key = abs)
corr_matrix.head(10)
```

[40]:

	SalePrice	MSSubClass	LotFrontage	LotArea	OverallQual	\
SalePrice	1.000000	-0.093480	0.318156	0.269070	0.804035	
OverallQual	0.804035	0.037913	0.239457	0.113560	1.000000	
GrLivArea	0.693735	0.058138	0.350559	0.278004	0.598385	
GarageCars	0.680259	-0.054044	0.280197	0.156982	0.581545	

```
GarageArea         0.643901   -0.117231     0.338159  0.182415     0.541389
FullBath           0.611537    0.125495     0.170479  0.134447     0.562303
TotalBsmtSF        0.592936   -0.243096     0.371004  0.269160     0.526140
ExterQual_TA      -0.588897   -0.067985    -0.123446 -0.019599    -0.638127
1stFlrSF           0.586409   -0.265793     0.421516  0.311075     0.462571
GarageFinish_Unf  -0.559765    0.046513    -0.236872 -0.084209    -0.512063

                  OverallCond  MasVnrArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  \
SalePrice           -0.066027    0.432571    0.366188    0.002978   0.218008
OverallQual         -0.113034    0.425914    0.236122   -0.061214   0.311266
GrLivArea           -0.095342    0.392901    0.231336   -0.026904   0.224774
GarageCars          -0.208628    0.370336    0.226537   -0.046619   0.208297
GarageArea          -0.184285    0.390140    0.316417   -0.021788   0.172652
FullBath            -0.247570    0.291096    0.102262   -0.119473   0.272438
TotalBsmtSF         -0.182606    0.387219    0.543963    0.118473   0.402793
ExterQual_TA         0.200865   -0.266009   -0.137986    0.081234  -0.266686
1stFlrSF            -0.153199    0.369777    0.459641    0.105827   0.297402
GarageFinish_Unf     0.229698   -0.285009   -0.233680    0.018388  -0.133172

                  …  GarageType_Detchd  GarageFinish_RFn  GarageFinish_Unf  \
SalePrice         …          -0.388296          0.232743         -0.559765
OverallQual       …          -0.325682          0.212133         -0.512063
GrLivArea         …          -0.213304          0.061485         -0.300139
GarageCars        …          -0.163182          0.205035         -0.468751
GarageArea        …          -0.141283          0.232154         -0.428303
FullBath          …          -0.300496          0.216216         -0.440858
TotalBsmtSF       …          -0.309600          0.151826         -0.365094
ExterQual_TA      …           0.287264         -0.241443          0.502316
1stFlrSF          …          -0.303681          0.109581         -0.317897
GarageFinish_Unf  …           0.556622         -0.612487          1.000000

                  SaleCondition_AdjLand  SaleCondition_Alloca  \
SalePrice                     -0.057542             -0.001620
OverallQual                   -0.036826             -0.041103
GrLivArea                     -0.038034              0.040631
GarageCars                    -0.108423              0.028389
GarageArea                    -0.100421             -0.009566
FullBath                       0.035945              0.030797
TotalBsmtSF                   -0.058123             -0.053870
ExterQual_TA                   0.035219              0.047855
1stFlrSF                      -0.011851              0.049135
GarageFinish_Unf               0.048368              0.074730

                  SaleCondition_Family  SaleCondition_Normal  \
SalePrice                    -0.046956             -0.122841
OverallQual                  -0.017457             -0.151060
GrLivArea                    -0.005037             -0.106393
```

```
GarageCars                     0.016485           -0.139635
GarageArea                     0.005757           -0.151301
FullBath                       0.038534           -0.171229
TotalBsmtSF                    0.025166           -0.176096
ExterQual_TA                   0.046967            0.148439
1stFlrSF                       0.032487           -0.184729
GarageFinish_Unf               0.002622            0.094083

                  SaleCondition_Partial  AgeSinceConst  AgeSinceRemod
SalePrice                      0.335221      -0.559540      -0.545044
OverallQual                    0.329897      -0.534620      -0.530012
GrLivArea                      0.168146      -0.177248      -0.265157
GarageCars                     0.297339      -0.529766      -0.412939
GarageArea                     0.309886      -0.469877      -0.359158
FullBath                       0.270859      -0.491070      -0.424726
TotalBsmtSF                    0.287225      -0.380173      -0.273979
ExterQual_TA                  -0.327772       0.584044       0.571001
1stFlrSF                       0.242419      -0.262939      -0.223534
GarageFinish_Unf              -0.254394       0.618939       0.438339

[10 rows x 135 columns]
```

Aqui é possível perceber que 'SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars' se destacam na correlação com a nossa variável dependente. Isso é importante na decisão das variáveis a serem adicionadas no modelo.

2.8 Outlier detection

[41]:
```python
n = 2

upper_limit = train['SalePrice'].mean() + n*train['SalePrice'].std()
lower_limit = train['SalePrice'].mean() - n*train['SalePrice'].std()

print("Highest allowed", upper_limit)
print("Lowest allowed", lower_limit)
```

```
Highest allowed 12.798646944221764
Lowest allowed 11.26664340018648
```

[42]:
```python
train_trimmed = train[
    ( train['SalePrice'] < upper_limit ) &

    ( train['SalePrice'] > lower_limit )
    ]
```

[43]:
```python
train.shape
```

[43]: (1022, 135)

[44]:
```
train_trimmed.shape #removeu 49 observações
```

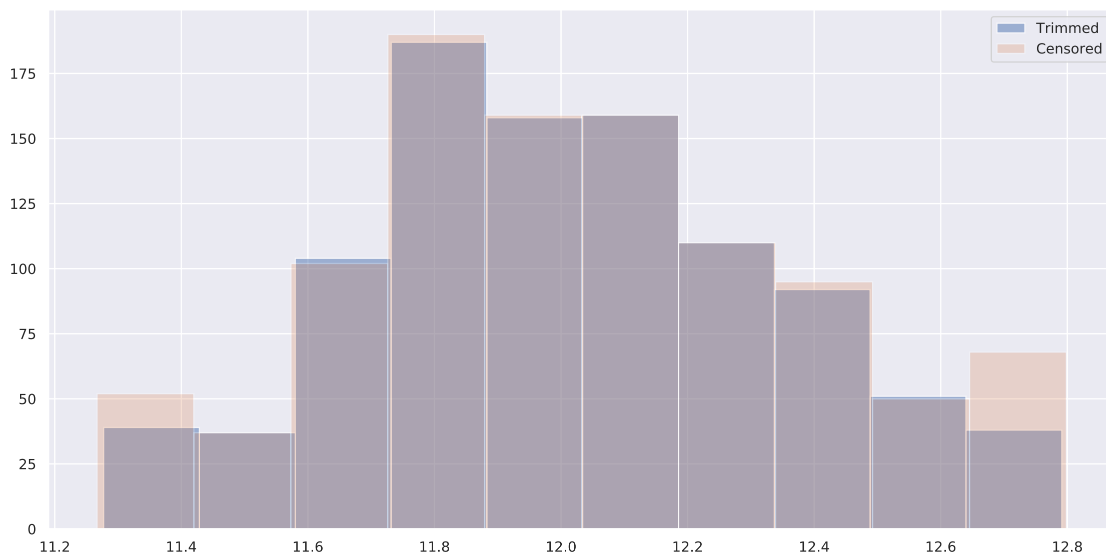
[44]: (975, 135)

[45]:
```
train_censored = pd.DataFrame()
train_censored['SalePrice'] =  np.where(
    train['SalePrice'] > upper_limit,
        upper_limit,
        np.where(
            train['SalePrice'] < lower_limit,
            lower_limit,
            train['SalePrice']
            )
        )
```

[46]:
```
train_censored.shape
```

[46]: (1022, 1)

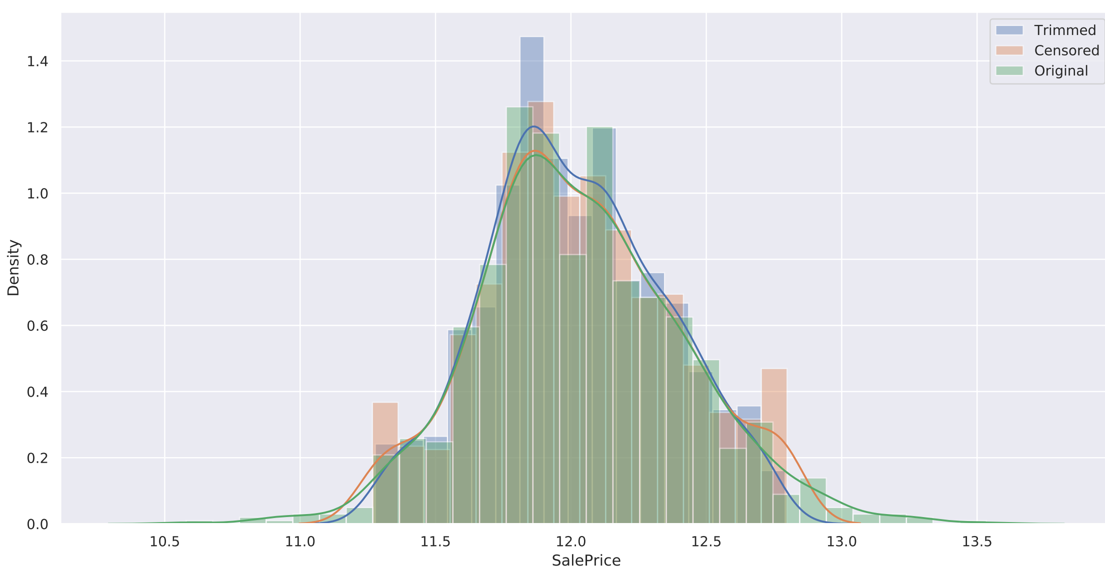
[47]:
```
plt.figure(figsize=(15, 7.5))
#plt.hist(train['SalePrice'], alpha=1, label='Original Data')
plt.hist(train_trimmed['SalePrice'], alpha=0.5, label='Trimmed')
plt.hist(train_censored['SalePrice'], alpha=0.25, label='Censored')

plt.legend(loc='upper right')
plt.show()
```



[48]:

```
plt.figure(figsize=(15, 7.5))
sns.distplot(train_trimmed['SalePrice'], kde = True, label='Trimmed');
sns.distplot(train_censored['SalePrice'], kde = True, label='Censored');
sns.distplot(train['SalePrice'], kde = True, label='Original')

plt.legend(loc='upper right')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



Optei por aplicar a regra de trimming por ter retirado algumas observações que estavam muito na ponta da distribuição

[49]:
```
train_trimmed
```

[49]:
```
      SalePrice  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  \
632   11.320566          20    85.000000    11900            7            5
208   12.531776          60    70.049958    14364            7            5
83    11.748005          20    80.000000     8892            5            5
1174  12.384223          70    80.000000    16560            6            8
895   11.849405          60    71.000000     7056            6            5
…           …         …           …       …          …           …
715   12.013707          20    78.000000    10140            6            5
905   11.759793          20    80.000000     9920            5            5
1096  11.751950          70    60.000000     6882            6            7
235   11.402005         160    21.000000     1680            6            3
1061  11.302217          30   120.000000    18000            3            4

      MasVnrArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  …  GarageType_Detchd  \
632        209.0         822           0        564  …                  0
208        128.0        1065           0         92  …                  0
83          66.0           0           0       1065  …                  1
1174         0.0         503           0        449  …                  1
895        415.0         400           0        380  …                  0
…            …         …         …         …  …                …
715        174.0           0           0       1064  …                  0
905        110.0         354         290        412  …                  0
1096         0.0           0           0        684  …                  0
235        604.0         358           0        125  …                  1
1061         0.0           0           0        894  …                  1

      GarageFinish_RFn  GarageFinish_Unf  SaleCondition_AdjLand  \
632                  0                 0                      0
208                  0                 0                      0
83                   0                 1                      0
1174                 0                 1                      0
895                  1                 0                      0
…                  …               …                    …
715                  1                 0                      0
905                  1                 0                      0
1096                 0                 1                      0
235                  0                 1                      0
1061                 1                 0                      0

      SaleCondition_Alloca  SaleCondition_Family  SaleCondition_Normal  \
632                      0                     1                     0
208                      0                     0                     1
83                       0                     0                     1
1174                     0                     0                     1
895                      0                     0                     1
```

```
…                    …                    …                    …
715                        0                     0                      1
905                        0                     0                      1
1096                       0                     0                      1
235                        0                     0                      1
1061                       0                     0                      1

      SaleCondition_Partial  AgeSinceConst  AgeSinceRemod
632                       0             33             33
208                       0             22             21
83                        0             50             50
1174                      0             78             60
895                       0             47             47
…                       …             …             …
715                       0             36             36
905                       0             56             56
1096                      0             96              4
235                       0             39             39
1061                      0             75             60

[975 rows x 135 columns]
```

[50]:
```
corr_matrix = train_trimmed.corr().sort_values(
    by = "SalePrice",  ascending=False, key = abs)
corr_matrix.head(10)
```

[50]:
```
                  SalePrice  MSSubClass  LotFrontage   LotArea  OverallQual  \
SalePrice          1.000000   -0.079544     0.297758  0.261926     0.765242
OverallQual        0.765242    0.068494     0.196234  0.083181     1.000000
GrLivArea          0.652940    0.076908     0.345469  0.298294     0.546127
GarageCars         0.631148   -0.031611     0.258319  0.154039     0.518169
FullBath           0.603836    0.136811     0.149749  0.119613     0.544895
ExterQual_TA      -0.602824   -0.082463    -0.102374 -0.017141    -0.641741
GarageArea         0.588006   -0.098682     0.322014  0.196633     0.470661
ExterQual_Gd       0.565698    0.069173     0.055267 -0.004558     0.584266
GarageFinish_Unf  -0.559319    0.035448    -0.221174 -0.063525    -0.495734
AgeSinceConst     -0.555052   -0.045496    -0.100923  0.007903    -0.519014

                  OverallCond  MasVnrArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  \
SalePrice           -0.103153    0.334670    0.288469   -0.002923   0.214769
OverallQual         -0.141659    0.349202    0.154586   -0.069117   0.306059
GrLivArea           -0.126640    0.303910    0.183064   -0.026485   0.206174
GarageCars          -0.243328    0.300903    0.161430   -0.045245   0.193314
FullBath            -0.283695    0.241421    0.059342   -0.125381   0.272561
ExterQual_TA         0.204201   -0.219349   -0.091613    0.076064  -0.270700
GarageArea          -0.216105    0.316116    0.263666   -0.013711   0.156801
ExterQual_Gd        -0.202342    0.171001    0.059425   -0.074602   0.242472
```

```
GarageFinish_Unf      0.240175   -0.258584   -0.195287    0.022165  -0.127460
AgeSinceConst         0.431220   -0.287606   -0.220009    0.062802  -0.128101

                  …  GarageType_Detchd  GarageFinish_RFn  GarageFinish_Unf  \
SalePrice         …          -0.405890          0.262486         -0.559319
OverallQual       …          -0.323482          0.235122         -0.495734
GrLivArea         …          -0.200777          0.070825         -0.274192
GarageCars        …          -0.147265          0.213703         -0.442904
FullBath          …          -0.289458          0.219449         -0.426703
ExterQual_TA      …           0.272961         -0.254440          0.490213
GarageArea        …          -0.121668          0.243049         -0.404039
ExterQual_Gd      …          -0.267917          0.271479         -0.474785
GarageFinish_Unf  …           0.554142         -0.626342          1.000000
AgeSinceConst     …           0.483393         -0.333466          0.601887

                  SaleCondition_AdjLand  SaleCondition_Alloca  \
SalePrice                     -0.066742              0.002584
OverallQual                   -0.039270             -0.042205
GrLivArea                     -0.039802              0.048364
GarageCars                    -0.114878              0.032963
FullBath                       0.038393              0.034061
ExterQual_TA                   0.034917              0.046531
GarageArea                    -0.107176             -0.007417
ExterQual_Gd                  -0.033250             -0.042865
GarageFinish_Unf               0.049071              0.075637
AgeSinceConst                  0.009299              0.007940

                  SaleCondition_Family  SaleCondition_Normal  \
SalePrice                    -0.050101             -0.041188
OverallQual                  -0.014436             -0.085902
GrLivArea                    -0.000287             -0.072248
GarageCars                    0.021165             -0.088508
FullBath                      0.042874             -0.150889
ExterQual_TA                  0.044554              0.111940
GarageArea                    0.010262             -0.091235
ExterQual_Gd                 -0.039250             -0.086990
GarageFinish_Unf              0.001543              0.064038
AgeSinceConst                 0.013576              0.141673

                  SaleCondition_Partial  AgeSinceConst  AgeSinceRemod
SalePrice                      0.232033      -0.555052      -0.522927
OverallQual                    0.254640      -0.519014      -0.502385
GrLivArea                      0.108802      -0.138068      -0.220205
GarageCars                     0.233804      -0.510475      -0.372464
FullBath                       0.241859      -0.483250      -0.405028
ExterQual_TA                  -0.287485       0.576681       0.558312
GarageArea                     0.235900      -0.448016      -0.314380
```

```
ExterQual_Gd                       0.243714      -0.570249      -0.541986
GarageFinish_Unf                  -0.222178       0.601887       0.415002
AgeSinceConst                     -0.313330       1.000000       0.564380

[10 rows x 135 columns]
```

É possível notar que, após removermos os outliers pelo método de trimming, a matriz de correlação sofre alterações. Por exemplo, GarageArea perde espaço para FullBath e ExterQual_TA.

3 Parte II: Escolhendo as variáveis para treinar o modelo

Primeiro, irei selecionar um subconjunto de features baseada na correlação das variáveis com SalePrice.

[51]:
```
# Agora vamos pegar um subconjunto com as features de maior correlação
# esse código também foi pego do kaggle

data_num_corr = train_trimmed.corr()["SalePrice"][:-1]

# Variáveis correlacionadas (r2 > 0.5)
high_features_list = data_num_corr[abs(data_num_corr) >= 0.5].
 ↪sort_values(ascending=False)
print(f"{len(high_features_list)} variáveis fortemente correlacionadas à␣
 ↪SalePrice:\n{high_features_list}\n")

# Variáveis correlacionadas (0.3 < r2 < 0.5)
low_features_list = data_num_corr[(abs(data_num_corr) < 0.5) &␣
 ↪(abs(data_num_corr) >= 0.3)].sort_values(ascending=False)
print(f"{len(low_features_list)} variáveis mediamente correlacionadas à␣
 ↪SalePrice:\n{low_features_list}")
```

```
15 variáveis fortemente correlacionadas à SalePrice:
SalePrice           1.000000
OverallQual         0.765242
GrLivArea           0.652940
GarageCars          0.631148
FullBath            0.603836
GarageArea          0.588006
ExterQual_Gd        0.565698
TotalBsmtSF         0.522193
Foundation_PConc    0.518911
1stFlrSF            0.507021
BsmtQual_TA        -0.541602
KitchenQual_TA     -0.545076
AgeSinceConst      -0.555052
GarageFinish_Unf   -0.559319
```

```
ExterQual_TA       -0.602824
Name: SalePrice, dtype: float64

15 variáveis mediamente correlacionadas à SalePrice:
TotRmsAbvGrd            0.472853
KitchenQual_Gd          0.467660
BsmtQual_Gd             0.448130
Fireplaces              0.439166
BsmtFinType1_GLQ        0.400416
Exterior1st_VinylSd     0.340144
MasVnrArea              0.334670
2ndFlrSF                0.323748
GarageType_Attchd       0.317285
OpenPorchSF             0.309191
GarageYrBlt             0.301843
LotShape_Reg           -0.309337
Foundation_CBlock      -0.337382
HeatingQC_TA           -0.338262
GarageType_Detchd      -0.405890
Name: SalePrice, dtype: float64
```

[52]:
```python
strong_features = data_num_corr[abs(data_num_corr) >= 0.5].index.tolist()
low_features = data_num_corr[(abs(data_num_corr) >= 0.3) & (abs(data_num_corr)␣
↪< 0.5)].index.tolist()
selected_features = strong_features[:-1] + low_features
```

[53]:
```python
selected_features
```

[53]:
```
['SalePrice',
 'OverallQual',
 'TotalBsmtSF',
 '1stFlrSF',
 'GrLivArea',
 'FullBath',
 'GarageCars',
 'GarageArea',
 'ExterQual_Gd',
 'ExterQual_TA',
 'Foundation_PConc',
 'BsmtQual_TA',
 'KitchenQual_TA',
 'GarageFinish_Unf',
 'MasVnrArea',
 '2ndFlrSF',
 'TotRmsAbvGrd',
 'Fireplaces',
 'GarageYrBlt',
```

```
 'OpenPorchSF',
 'LotShape_Reg',
 'Exterior1st_VinylSd',
 'Foundation_CBlock',
 'BsmtQual_Gd',
 'BsmtFinType1_GLQ',
 'HeatingQC_TA',
 'KitchenQual_Gd',
 'GarageType_Attchd',
 'GarageType_Detchd']
```

[54]:
```
corr_matrix = train_trimmed[selected_features].corr().sort_values(
    by = "SalePrice",  ascending=False, key = abs)
corr_matrix.head(10)
```

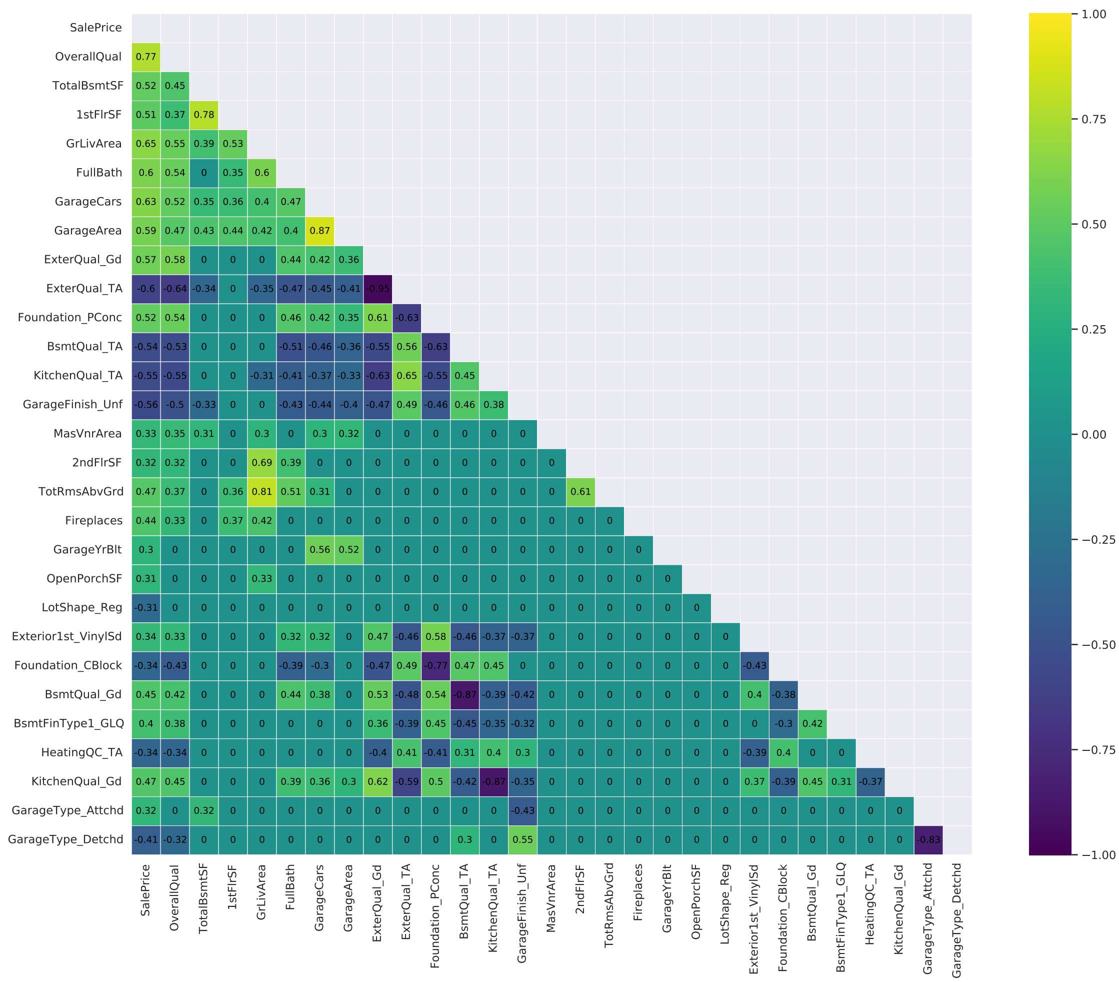
[54]:
```
               SalePrice  OverallQual  TotalBsmtSF  1stFlrSF  GrLivArea  \
SalePrice       1.000000     0.765242     0.522193  0.507021   0.652940
OverallQual     0.765242     1.000000     0.447859  0.369176   0.546127
GrLivArea       0.652940     0.546127     0.392906  0.529928   1.000000
GarageCars      0.631148     0.518169     0.349583  0.364132   0.404104
FullBath        0.603836     0.544895     0.291052  0.345608   0.597319
ExterQual_TA   -0.602824    -0.641741    -0.342623 -0.235157  -0.346735
GarageArea      0.588006     0.470661     0.432695  0.439012   0.416010
ExterQual_Gd    0.565698     0.584266     0.280458  0.182351   0.289482
GarageFinish_Unf -0.559319  -0.495734    -0.327293 -0.279456  -0.274192
KitchenQual_TA -0.545076    -0.550811    -0.273059 -0.220708  -0.311830

               FullBath  GarageCars  GarageArea  ExterQual_Gd  \
SalePrice      0.603836    0.631148    0.588006      0.565698
OverallQual    0.544895    0.518169    0.470661      0.584266
GrLivArea      0.597319    0.404104    0.416010      0.289482
GarageCars     0.470421    1.000000    0.871480      0.424618
FullBath       1.000000    0.470421    0.397575      0.442255
ExterQual_TA  -0.471798   -0.451159   -0.406195     -0.952259
GarageArea     0.397575    0.871480    1.000000      0.362880
ExterQual_Gd   0.442255    0.424618    0.362880      1.000000
GarageFinish_Unf -0.426703 -0.442904  -0.404039     -0.474785
KitchenQual_TA -0.412704   -0.373925   -0.331929     -0.628346

               ExterQual_TA  …  OpenPorchSF  LotShape_Reg  \
SalePrice         -0.602824  …     0.309191     -0.309337
OverallQual       -0.641741  …     0.275481     -0.169979
GrLivArea         -0.346735  …     0.334865     -0.166017
GarageCars        -0.451159  …     0.165746     -0.196596
FullBath          -0.471798  …     0.267460     -0.136222
ExterQual_TA       1.000000  …    -0.219772      0.173057
GarageArea        -0.406195  …     0.199279     -0.178541
```

```
ExterQual_Gd        -0.952259  …      0.167587      -0.166957
GarageFinish_Unf     0.490213  …     -0.182094       0.210431
KitchenQual_TA       0.645749  …     -0.187685       0.128262

                  Exterior1st_VinylSd  Foundation_CBlock  BsmtQual_Gd  \
SalePrice                    0.340144          -0.337382     0.448130
OverallQual                  0.332391          -0.428715     0.418217
GrLivArea                    0.098834          -0.245454     0.178350
GarageCars                   0.323036          -0.302779     0.378159
FullBath                     0.321020          -0.392988     0.436926
ExterQual_TA                -0.455324           0.487397    -0.483616
GarageArea                   0.257297          -0.230435     0.270798
ExterQual_Gd                 0.465977          -0.469645     0.525142
GarageFinish_Unf            -0.368085           0.237904    -0.420606
KitchenQual_TA              -0.373829           0.452871    -0.389313

                  BsmtFinType1_GLQ  HeatingQC_TA  KitchenQual_Gd  \
SalePrice                 0.400416     -0.338262        0.467660
OverallQual               0.381896     -0.336398        0.446987
GrLivArea                 0.133766     -0.163177        0.232349
GarageCars                0.279267     -0.216852        0.360678
FullBath                  0.250541     -0.256587        0.389656
ExterQual_TA             -0.388054      0.406962       -0.588198
GarageArea                0.270306     -0.138331        0.300002
ExterQual_Gd              0.361606     -0.395953        0.615329
GarageFinish_Unf         -0.322649      0.301495       -0.351860
KitchenQual_TA           -0.350440      0.396040       -0.870662

                  GarageType_Attchd  GarageType_Detchd
SalePrice                  0.317285          -0.405890
OverallQual                0.263570          -0.323482
GrLivArea                  0.064534          -0.200777
GarageCars                 0.062236          -0.147265
FullBath                   0.185131          -0.289458
ExterQual_TA              -0.205785           0.272961
GarageArea                 0.056247          -0.121668
ExterQual_Gd               0.210497          -0.267917
GarageFinish_Unf          -0.434140           0.554142
KitchenQual_TA            -0.153015           0.201672

[10 rows x 29 columns]
```

[55]:
```
pd.options.display.float_format = "{:,.2f}".format

# Define correlation matrix
corr_matrix = train_trimmed[selected_features].corr()
```

```python
# Replace any correlation < |0.3| by 0 for a better visibility
corr_matrix[(corr_matrix < 0.3) & (corr_matrix > -0.3)] = 0

# Mask the upper part of the heatmap
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Choose the color map
cmap = "viridis"

# plot the heatmap
sns.set(rc = {'figure.figsize':(20,15)})
sns.heatmap(corr_matrix, mask=mask, vmax=1.0, vmin=-1.0, linewidths=0.1,
            annot_kws={"size": 9, "color": "black"}, square=True, cmap=cmap,␣
 ↪annot=True)
```

[55]: <matplotlib.axes._subplots.AxesSubplot at 0x7f06f5518b50>



4 Parte III: Modelos e acurácia

[56]:
```
train_sub = train_trimmed[selected_features]
test_sub = test[selected_features]
```

[57]:
```
x_train = train_sub.drop(['SalePrice'], 1)
y_train = train_sub['SalePrice']

x_test = test_sub.drop(['SalePrice'], 1)
y_test = test['SalePrice']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  after removing the cwd from sys.path.
```

[58]:
```
# irei usar duas métricas de acurácia: mae e mape

from sklearn.metrics import mean_absolute_error as mae

def mape(Y_actual, Y_Predicted):
    mape = (np.mean(np.abs(Y_actual - Y_Predicted)/Y_actual))*100
    return mape
```

[59]:
```
naive_model = y_train.mean()
pred_naive = np.repeat(naive_model, len(y_test))

print('Mape: ', round(mape(np.exp(y_test) , np.exp(pred_naive)), 2))
print('Mae: ', round(mae(np.exp(y_test) , np.exp(pred_naive)), 2))
```

```
Mape:  36.07
Mae:  58402.93
```

LINEAR

[60]:
```
X_lr = sm.add_constant(x_train)

X_lr_test = sm.add_constant(x_test)
linear_reg = sm.OLS(y_train, X_lr )
linear_reg_fit  = linear_reg.fit()
linear_reg_pred = linear_reg_fit.predict(X_lr_test)
print(linear_reg_fit.summary())

print('Mape')
```

```
print(round(mape(np.exp(y_test) , np.exp(linear_reg_pred)), 2))

print('Mae')
print(round(mae(np.exp(y_test) , np.exp(linear_reg_pred)), 2))
```

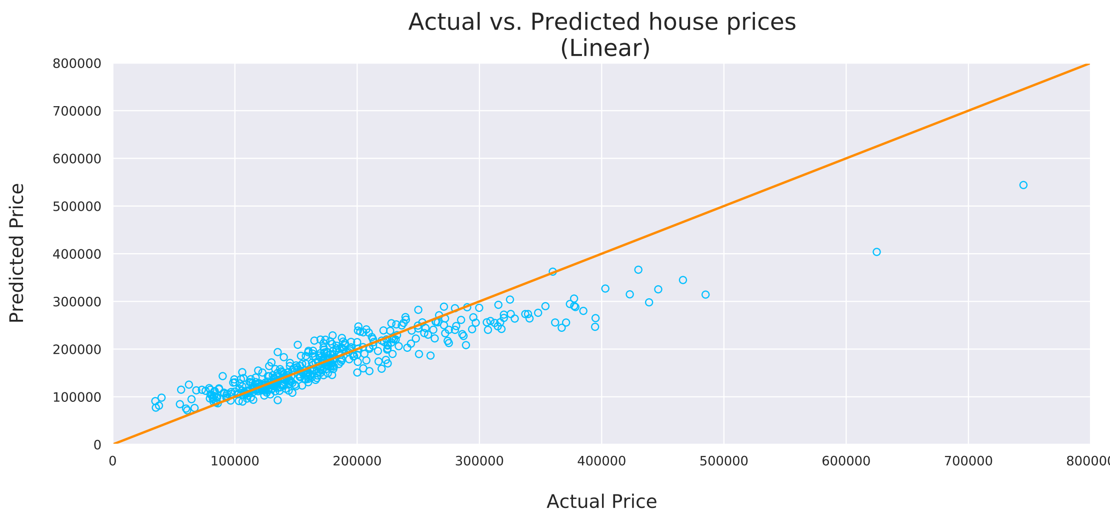
```
                            OLS Regression Results
==============================================================================
Dep. Variable:              SalePrice   R-squared:                       0.807
Model:                            OLS   Adj. R-squared:                  0.801
Method:                 Least Squares   F-statistic:                     141.4
Date:                Sun, 15 May 2022   Prob (F-statistic):          6.33e-315
Time:                        18:43:17   Log-Likelihood:                 509.03
No. Observations:                 975   AIC:                            -960.1
Df Residuals:                     946   BIC:                            -818.5
Df Model:                          28
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
const                  10.8576      0.070    155.841      0.000      10.721
10.994
OverallQual             0.0701      0.006     11.102      0.000       0.058
0.083
TotalBsmtSF          3.907e-05   2.03e-05      1.926      0.054   -7.38e-07
7.89e-05
1stFlrSF                0.0002      0.000      1.472      0.141   -5.36e-05
0.000
GrLivArea           -1.715e-05      0.000     -0.160      0.873      -0.000
0.000
FullBath                0.0285      0.013      2.114      0.035       0.002
0.055
GarageCars              0.0536      0.016      3.326      0.001       0.022
0.085
GarageArea           6.951e-05   5.31e-05      1.309      0.191   -3.47e-05
0.000
ExterQual_Gd            0.0920      0.036      2.567      0.010       0.022
0.162
ExterQual_TA            0.0646      0.037      1.758      0.079      -0.008
0.137
Foundation_PConc        0.0676      0.020      3.336      0.001       0.028
0.107
BsmtQual_TA            -0.0150      0.023     -0.659      0.510      -0.060
0.030
KitchenQual_TA         -0.0683      0.022     -3.078      0.002      -0.112
```

```
-0.025
GarageFinish_Unf       -0.0221      0.014     -1.592      0.112      -0.049
0.005
MasVnrArea          -5.914e-05   3.41e-05     -1.734      0.083      -0.000
7.78e-06
2ndFlrSF                0.0001      0.000      1.331      0.184   -6.81e-05
0.000
TotRmsAbvGrd            0.0150      0.006      2.706      0.007       0.004
0.026
Fireplaces              0.0564      0.009      6.474      0.000       0.039
0.074
GarageYrBlt          5.282e-05   1.61e-05      3.273      0.001    2.11e-05
8.45e-05
OpenPorchSF             0.0001   7.74e-05      1.700      0.090   -2.03e-05
0.000
LotShape_Reg           -0.0594      0.010     -5.803      0.000      -0.080
-0.039
Exterior1st_VinylSd     0.0075      0.013      0.584      0.560      -0.018
0.033
Foundation_CBlock       0.0751      0.017      4.379      0.000       0.041
0.109
BsmtQual_Gd             0.0099      0.021      0.461      0.645      -0.032
0.052
BsmtFinType1_GLQ        0.0513      0.012      4.124      0.000       0.027
0.076
HeatingQC_TA           -0.0411      0.012     -3.378      0.001      -0.065
-0.017
KitchenQual_Gd         -0.0239      0.022     -1.109      0.268      -0.066
0.018
GarageType_Attchd       0.0538      0.019      2.881      0.004       0.017
0.090
GarageType_Detchd      -0.0036      0.022     -0.167      0.868      -0.047
0.039
==============================================================================
Omnibus:                      453.973   Durbin-Watson:                   2.036
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8675.146
Skew:                          -1.655   Prob(JB):                         0.00
Kurtosis:                      17.233   Cond. No.                     4.88e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.88e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
Mape
13.74
Mae
```

23627.56

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:117:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[61]:
```python
plt.figure(figsize=(15,6))
plt.title("Actual vs. Predicted house prices\n (Linear)", fontsize=20)
plt.scatter(np.exp(y_test), np.exp(linear_reg_pred),
            color="deepskyblue", marker="o", facecolors="none")
plt.plot([0, 800000], [0, 800000], "darkorange", lw=2)
plt.xlim(0, 800000)
plt.ylim(0, 800000)
plt.xlabel("\nActual Price", fontsize=16)
plt.ylabel("Predicted Price\n", fontsize=16)
plt.show()
```

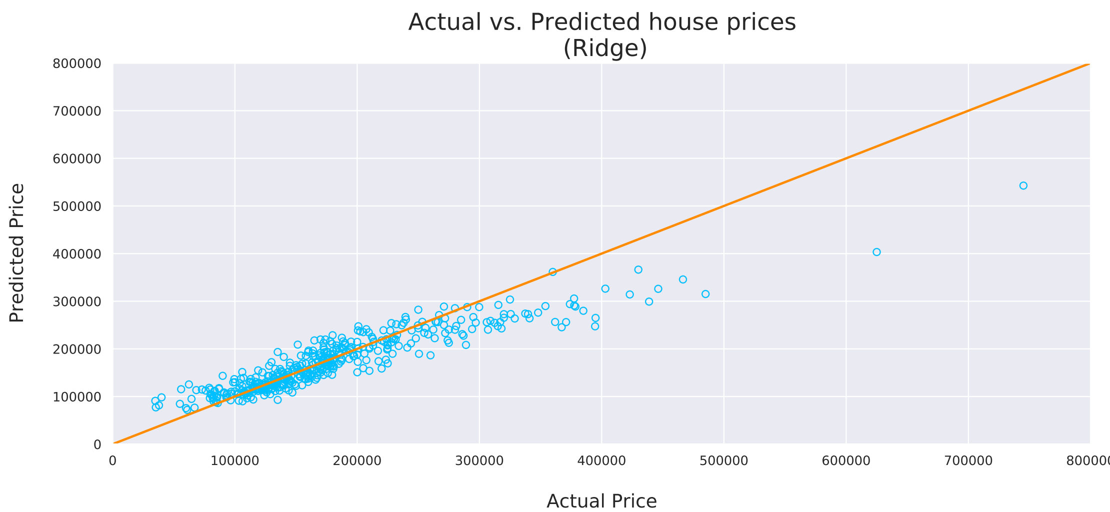


RIDGE

[62]:
```python
ridge_reg = Ridge(alpha = 0.5)
ridge_reg.fit(x_train, y_train)
ridge_pred = ridge_reg.predict(x_test)

print(round(mape(np.exp(y_test), np.exp(ridge_pred)),2))
print(round(mae(np.exp(y_test), np.exp(ridge_pred)), 2))
```

13.74
23622.35

[63]:
```
plt.figure(figsize=(15,6))
plt.title("Actual vs. Predicted house prices\n (Ridge)", fontsize=20)
plt.scatter(np.exp(y_test), np.exp(ridge_pred),
            color="deepskyblue", marker="o", facecolors="none")
plt.plot([0, 800000], [0, 800000], "darkorange", lw=2)
plt.xlim(0, 800000)
plt.ylim(0, 800000)
plt.xlabel("\nActual Price", fontsize=16)
plt.ylabel("Predicted Price\n", fontsize=16)
plt.show()
```



[64]:
```
# Procurando o melhor hiperparâmtero

alphas = np.linspace(0, 10, 100).tolist()

tuned_parameters = {"alpha": alphas}

# GridSearch
ridge_cv = GridSearchCV(Ridge(), tuned_parameters, cv=10, n_jobs=-1, verbose=1)

# fit the GridSearch on train set
ridge_cv.fit(x_train, y_train)

# print best params
print(f"Best hyperparameters: {ridge_cv.best_params_}")
```

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
Best hyperparameters: {'alpha': 10.0}
```

[65]:
```
ridge_reg_h = Ridge(alpha = 10)
ridge_reg_h.fit(x_train, y_train)
ridge_pred_h = ridge_reg_h.predict(x_test)

print(round(mape(np.exp(y_test), np.exp(ridge_pred_h)), 2))
print(round(mae(np.exp(y_test), np.exp(ridge_pred_h)), 2))
```

```
13.79
23616.41
```

LASSO

[66]:
```
model_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(x_train, y_train)

lasso_pred = model_lasso.predict(x_test)

coef = pd.Series(model_lasso.coef_, index = x_train.columns)
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the␣
↪other " +  str(sum(coef == 0)) + " variables")
```

```
Lasso picked 27 variables and eliminated the other 1 variables
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.015720623839868608, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.03784634269239895, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 7.104e-01, tolerance: 1.042e-02
  coef_, l1_reg, l2_reg, X, y, max_iter, tol, rng, random, positive
```

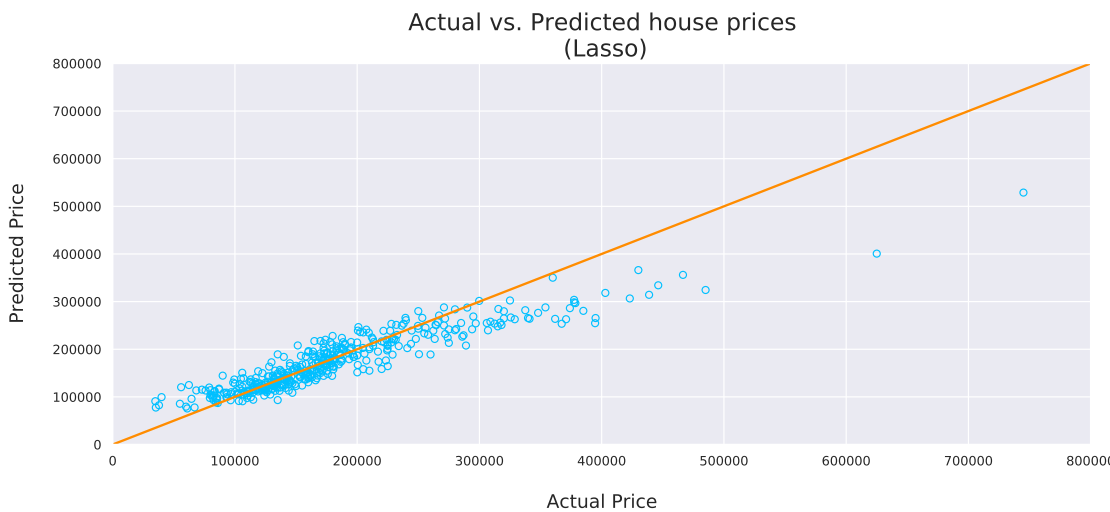
[67]:
```
imp_coef = pd.concat([coef.sort_values().head(10),
                     coef.sort_values().tail(10)])
plt.figure(figsize=(15,6))
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Lasso Model")
```

[67]:
```
Text(0.5, 1.0, 'Coefficients in the Lasso Model')
```

[68]:
```
print(round(mape(np.exp(y_test), np.exp(lasso_pred)), 2))
print(round(mae(np.exp(y_test), np.exp(lasso_pred)), 2))
```

```
13.8
23588.88
```

[69]:
```
plt.figure(figsize=(15,6))
plt.title("Actual vs. Predicted house prices\n (Lasso)", fontsize=20)
plt.scatter(np.exp(y_test), np.exp(lasso_pred),
            color="deepskyblue", marker="o", facecolors="none")
plt.plot([0, 800000], [0, 800000], "darkorange", lw=2)
plt.xlim(0, 800000)
plt.ylim(0, 800000)
plt.xlabel("\nActual Price", fontsize=16)
plt.ylabel("Predicted Price\n", fontsize=16)
plt.show()
```



ELASTIC NET

[70]:
```
model_ElasticNet = ElasticNetCV(
    l1_ratio = [.1, .5, .7, .9, .95, .99, 1],
    alphas = [1, 0.1, 0.001, 0.0005],
    fit_intercept = True
).fit(x_train, y_train)

elastic_pred = model_ElasticNet.predict(x_test)
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.009113857150918392, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 5.47604405957265, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.040674106959269096, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.01214585746170016, tolerance: 0.00850941007941276
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.026261637007873873, tolerance: 0.00821967321512788
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 4.057549191634133, tolerance: 0.00821967321512788
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.04444279017749153, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
```

```
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.17776310058462919, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.04159179426849491, tolerance: 0.00821967321512788
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.04073497675439697, tolerance: 0.00821967321512788
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.04830361854554255, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.06541470056892962, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.01445738086201942, tolerance: 0.00821967321512788
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.012795772561684515, tolerance: 0.00821967321512788
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.029517522335972046, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.04409889016523749, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.022379176302564474, tolerance: 0.00848991145135695
```

```
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.04073765252807071, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.017039309645582534, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.03839064338244391, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.015720623839868608, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:644: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 0.03784634269239895, tolerance: 0.00848991145135695
  positive,
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 3.787e+00, tolerance: 1.042e-02
  coef_, l1_reg, l2_reg, X, y, max_iter, tol, rng, random, positive
```

[71]:
```
coef = pd.Series(model_ElasticNet.coef_, index = x_train.columns)
print("Elastic Net picked " + str(sum(coef != 0)) + " variables and eliminated␣
↪the other " +  str(sum(coef == 0)) + " variables")
```

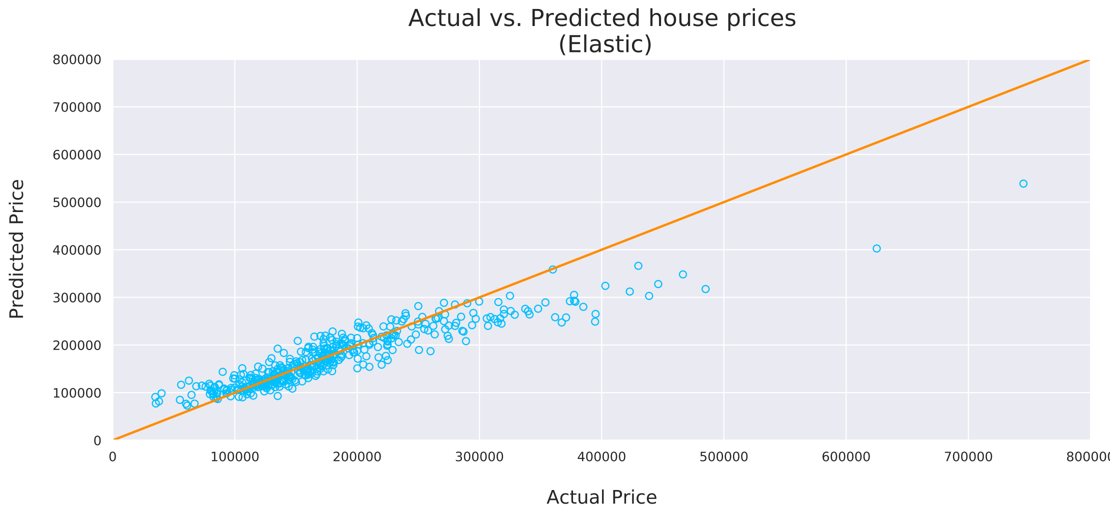
```
Elastic Net picked 28 variables and eliminated the other 0 variables
```

[72]:
```
print(round(mape(np.exp(y_test), np.exp(elastic_pred)), 2))
print(round(mae(np.exp(y_test), np.exp(elastic_pred)), 2))
```

```
13.75
23608.62
```

[73]:
```
plt.figure(figsize=(15,6))
plt.title("Actual vs. Predicted house prices\n (Elastic)", fontsize=20)
plt.scatter(np.exp(y_test), np.exp(elastic_pred),
```

```
             color="deepskyblue", marker="o", facecolors="none")
plt.plot([0, 800000], [0, 800000], "darkorange", lw=2)
plt.xlim(0, 800000)
plt.ylim(0, 800000)
plt.xlabel("\nActual Price", fontsize=16)
plt.ylabel("Predicted Price\n", fontsize=16)
plt.show()
```



Há uma redução considerável do modelo naive. Entretanto, não há uma melhora na acurácia do modelo mais simples de regressão linear para as outras regressões.

5 Exercício 3

6 Exercício 1 com Random Forest

[74]:
```
train_sub = train_trimmed[selected_features]
test_sub = test[selected_features]
```

[75]:
```
x_train = train_sub.drop(['SalePrice'], 1)
y_train = train_sub['SalePrice']

x_test = test_sub.drop(['SalePrice'], 1)
y_test = test['SalePrice']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  after removing the cwd from sys.path.
```

[76]:
```
parameters = {'max_depth' : [8, 10, 12, 20],
              'n_estimators' : [200, 250, 300],
              'max_features' : [5, 25, 50],
              'min_samples_split' : [2, 5, 10],
              'min_samples_leaf': [1, 2, 4]}
grid_search = GridSearchCV(estimator = RandomForestRegressor(random_state=42),
                           param_grid = parameters,
                           scoring = 'neg_mean_squared_error',
                           cv = 5)
grid_search.fit(x_train, y_train)
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
540 fits failed out of a total of 1620.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
540 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-
packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py",
line 467, in fit
    for i, t in enumerate(trees)
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 1043,
in __call__
    if self.dispatch_one_batch(iterator):
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 861, in
dispatch_one_batch
    self._dispatch(tasks)
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 779, in
_dispatch
    job = self._backend.apply_async(batch, callback=cb)
  File "/usr/local/lib/python3.7/dist-packages/joblib/_parallel_backends.py",
line 208, in apply_async
    result = ImmediateResult(func)
  File "/usr/local/lib/python3.7/dist-packages/joblib/_parallel_backends.py",
line 572, in __init__
```

```
    self.results = batch()
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 263, in
__call__
    for func, args, kwargs in self.items]
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 263, in
<listcomp>
    for func, args, kwargs in self.items]
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/fixes.py", line
216, in __call__
    return self.function(*args, **kwargs)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py",
line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line
1320, in fit
    X_idx_sorted=X_idx_sorted,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line
308, in fit
    raise ValueError("max_features must be in (0, n_features]")
ValueError: max_features must be in (0, n_features]

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972:
UserWarning: One or more of the test scores are non-finite: [-0.01896706
-0.01898649 -0.01897022 -0.01893804 -0.01892914 -0.01894571
 -0.01946076 -0.01935065 -0.01938406 -0.01905029 -0.01901579 -0.01912644
 -0.01920898 -0.0191056  -0.01913571 -0.01958489 -0.01954309 -0.01957918
 -0.01969193 -0.01966718 -0.01969125 -0.01969193 -0.01966718 -0.01969125
 -0.01986944 -0.0198002  -0.01987509 -0.01985531 -0.0198886  -0.01990123
 -0.01996483 -0.01999268 -0.02001972 -0.01998797 -0.0199928  -0.02002753
 -0.01979985 -0.01980799 -0.01985574 -0.01991794 -0.01989685 -0.01989192
 -0.01998808 -0.01995977 -0.01996378 -0.02012455 -0.02009982 -0.02010081
 -0.02012455 -0.02009982 -0.02010081 -0.02015976 -0.02013038 -0.02013622
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan -0.01873872 -0.01867607 -0.01864149
 -0.01876762 -0.0187078  -0.01871583 -0.01909134 -0.01901676 -0.01905131
 -0.01863874 -0.0186674  -0.01874125 -0.01892093 -0.0188608  -0.01888679
 -0.01917117 -0.01907235 -0.01916995 -0.01954061 -0.01947891 -0.01949421
 -0.01954061 -0.01947891 -0.01949421 -0.01976837 -0.0196858  -0.01965654
 -0.01995444 -0.01990562 -0.01992501 -0.0198037  -0.01982654 -0.01987877
 -0.01989572 -0.01988623 -0.01984846 -0.01972553 -0.01965365 -0.01970624
 -0.01979149 -0.01972679 -0.01977268 -0.01984003 -0.01980001 -0.01978947
 -0.02006911 -0.0200693  -0.02006919 -0.02006911 -0.0200693  -0.02006919
 -0.02006502 -0.02007266 -0.02009244         nan         nan         nan
         nan         nan         nan         nan         nan         nan
```

```
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
 -0.01832886 -0.01834482 -0.01832992 -0.01854961 -0.01855267 -0.01853636
 -0.01892466 -0.01892504 -0.01893053 -0.01860006 -0.01860433 -0.01866357
 -0.01859305 -0.01860361 -0.01871056 -0.01893515 -0.01894583 -0.01902127
 -0.0194898  -0.01936848 -0.01939574 -0.0194898  -0.01936848 -0.01939574
 -0.01964322 -0.01963457 -0.01961749 -0.01977759 -0.01976367 -0.01978619
 -0.01992553 -0.01988791 -0.01987138 -0.01989468 -0.01990796 -0.01990784
 -0.01962636 -0.01961479 -0.01967087 -0.01962065 -0.01956434 -0.01959725
 -0.01992592 -0.01989678 -0.01988595 -0.020011   -0.01998522 -0.0200036
 -0.020011   -0.01998522 -0.0200036  -0.02007978 -0.02007098 -0.02008165
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan -0.01837679 -0.01837134 -0.01837287
 -0.01854514 -0.01850958 -0.01851413 -0.01881869 -0.01887698 -0.01885746
 -0.01878273 -0.01869417 -0.01874594 -0.01875311 -0.01878044 -0.01881256
 -0.01889536 -0.01891449 -0.01897078 -0.01934366 -0.01927096 -0.01932423
 -0.01934366 -0.01927096 -0.01932423 -0.01965375 -0.01964522 -0.01964681
 -0.01982994 -0.01982474 -0.01980265 -0.01991682 -0.01992375 -0.01990894
 -0.01989653 -0.01989465 -0.01989887 -0.01961495 -0.0196507  -0.01965786
 -0.01968063 -0.01965615 -0.01971666 -0.01990684 -0.01985649 -0.01986462
 -0.02001798 -0.02000511 -0.02001405 -0.02001798 -0.02000511 -0.02001405
 -0.02001229 -0.02003779 -0.02002946         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan]
  category=UserWarning,
```

[76]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(random_state=42),
             param_grid={'max_depth': [8, 10, 12, 20],
                         'max_features': [5, 25, 50],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [200, 250, 300]},
             scoring='neg_mean_squared_error')
```

[77]:
```
grid_search.best_params_
```

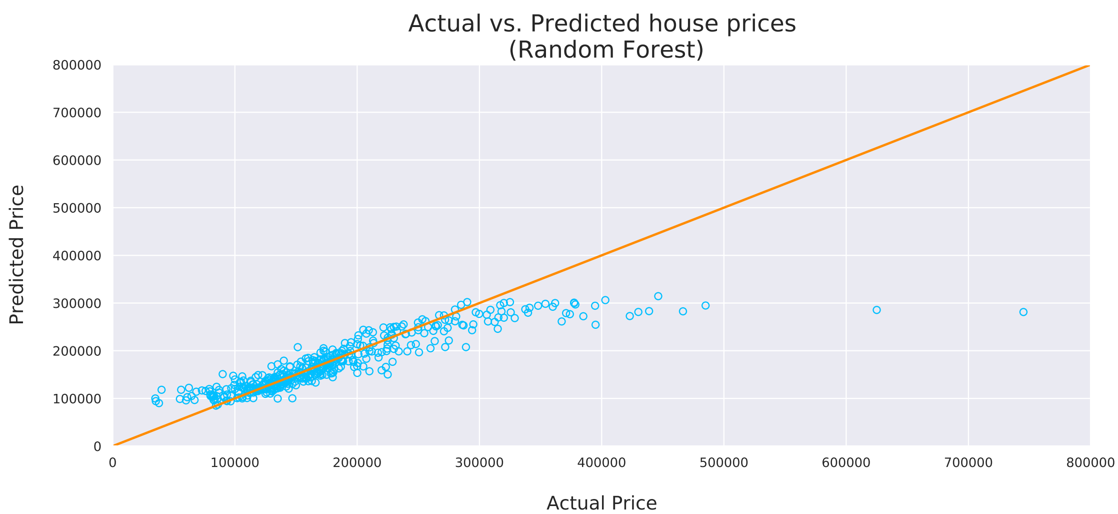
[77]:
```
{'max_depth': 12,
 'max_features': 5,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 200}
```

[78]:
```
regressor_rf = RandomForestRegressor(max_depth=12, n_estimators=200,␣
↪max_features=5, min_samples_split=2, min_samples_leaf = 1, random_state=1)
regressor_rf.fit(x_train, y_train)
rf_pred = regressor_rf.predict(x_test)
```

[79]:
```
print('Mape')
print(round(mape(np.exp(y_test) , np.exp(rf_pred)), 2))

print('Mae')
print(round(mae(np.exp(y_test) , np.exp(rf_pred)), 2))
```

```
Mape
13.74
Mae
23225.76
```

[80]:
```
plt.figure(figsize=(15,6))
plt.title("Actual vs. Predicted house prices\n (Random Forest)", fontsize=20)
plt.scatter(np.exp(y_test), np.exp(rf_pred),
            color="deepskyblue", marker="o", facecolors="none")
plt.plot([0, 800000], [0, 800000], "darkorange", lw=2)
plt.xlim(0, 800000)
plt.ylim(0, 800000)
plt.xlabel("\nActual Price", fontsize=16)
plt.ylabel("Predicted Price\n", fontsize=16)
plt.show()
```



O Mape não sofreu alterações, mas o MAE no modelo diminuiu quando usamos Random Forest.

7 Exercício 2

Na primeira parte do exercício 2 irei passar análise explanatória e pela preparação dos dados tanto para o dataset de mateméticq quanto para o de português.

Em suma:

- Variáveis binárias de 'yes' e 'no' são substituídas por 1 e 0;
- Variáveis categoricas são mantidas como de 1 a 5, isso pois há uma ordem de grandeza estabelecida então é possível manter os dados como estão;
- Variáveis que identificam a escola, o gênero, endereço e etc são tranformadas em dummies.

Na segunda parte, olho com mais atençao apenas para o consumo de alcool (Dalc e Walc) e como afeta G3.

Na terceira parte, analiso a importância das variáveis para G3 tanto pelo peso dos coeficientes de uma regressão linear quanto por random forest. Por curiosidade, faço isso considerando tanto o conjunto total dos dados como somente a parte de treino após o split.

Depois, analiso rapidamente como os dados se comportam dentro de diversos modelos vistos em aula.

[81]:
```
uploaded = files.upload()
mat = pd.read_csv(io.BytesIO(uploaded['student-mat.csv']))
por = pd.read_csv(io.BytesIO(uploaded['student-por.csv']))
```

```
<IPython.core.display.HTML object>


Saving student-mat.csv to student-mat (1).csv
Saving student-por.csv to student-por (1).csv
```

[82]:
```
# Tirando as variáveis G1 e G2 que devemos ignorar

mat=mat.drop(['G1', 'G2'], axis = 1)
por=por.drop(['G1', 'G2'], axis = 1)
```

8 Parte I: Conhecendo e preparando os dados

Aqui é possível vermos tanto o tamanho dos dataset quanto o tipo das variáveis com as quais estamos trabalhando. Observando os tipos das variáveis, é possível tomarmos decisões, como, por exemplo, tornar as variáveis em dummies.

Neste caso, temos diversas variáveis binárias que são objetos 'yes' ou 'no': schoolsup, famsup, paid, activities, nursery, higher, internet e romantic.

Enquanto outras variáveis do tipo int64 represetam escalas de 'very low' a 'very high': famrel, freetime, goout, Dalc, Walc e health. Como há uma ordem lógica de grandeza nestas variáveis, não é necessário mudá-las. Caso não houvesse essa ordem, seria necessário fazer a transformação em dummies.

Por fim, as variáveis 'school', 'address', 'famsize' e 'Pstatus' são strings que tem 2 possíveis observações e, assim, podem também ser transformadas em dummies.

[83]:
```
mat.describe(include='all').T
```

[83]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
school	395	2	GP	349	NaN	NaN	NaN	NaN	NaN	NaN	NaN
sex	395	2	F	208	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	395.00	NaN	NaN	NaN	16.70	1.28	15.00	16.00	17.00	18.00	22.00
address	395	2	U	307	NaN	NaN	NaN	NaN	NaN	NaN	NaN
famsize	395	2	GT3	281	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Pstatus	395	2	T	354	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Medu	395.00	NaN	NaN	NaN	2.75	1.09	0.00	2.00	3.00	4.00	4.00
Fedu	395.00	NaN	NaN	NaN	2.52	1.09	0.00	2.00	2.00	3.00	4.00
Mjob	395	5	other	141	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Fjob	395	5	other	217	NaN	NaN	NaN	NaN	NaN	NaN	NaN
reason	395	4	course	145	NaN	NaN	NaN	NaN	NaN	NaN	NaN
guardian	395	3	mother	273	NaN	NaN	NaN	NaN	NaN	NaN	NaN
traveltime	395.00	NaN	NaN	NaN	1.45	0.70	1.00	1.00	1.00	2.00	4.00
studytime	395.00	NaN	NaN	NaN	2.04	0.84	1.00	1.00	2.00	2.00	4.00
failures	395.00	NaN	NaN	NaN	0.33	0.74	0.00	0.00	0.00	0.00	3.00
schoolsup	395	2	no	344	NaN	NaN	NaN	NaN	NaN	NaN	NaN
famsup	395	2	yes	242	NaN	NaN	NaN	NaN	NaN	NaN	NaN
paid	395	2	no	214	NaN	NaN	NaN	NaN	NaN	NaN	NaN
activities	395	2	yes	201	NaN	NaN	NaN	NaN	NaN	NaN	NaN
nursery	395	2	yes	314	NaN	NaN	NaN	NaN	NaN	NaN	NaN
higher	395	2	yes	375	NaN	NaN	NaN	NaN	NaN	NaN	NaN
internet	395	2	yes	329	NaN	NaN	NaN	NaN	NaN	NaN	NaN
romantic	395	2	no	263	NaN	NaN	NaN	NaN	NaN	NaN	NaN
famrel	395.00	NaN	NaN	NaN	3.94	0.90	1.00	4.00	4.00	5.00	5.00
freetime	395.00	NaN	NaN	NaN	3.24	1.00	1.00	3.00	3.00	4.00	5.00
goout	395.00	NaN	NaN	NaN	3.11	1.11	1.00	2.00	3.00	4.00	5.00
Dalc	395.00	NaN	NaN	NaN	1.48	0.89	1.00	1.00	1.00	2.00	5.00
Walc	395.00	NaN	NaN	NaN	2.29	1.29	1.00	1.00	2.00	3.00	5.00
health	395.00	NaN	NaN	NaN	3.55	1.39	1.00	3.00	4.00	5.00	5.00
absences	395.00	NaN	NaN	NaN	5.71	8.00	0.00	0.00	4.00	8.00	75.00
G3	395.00	NaN	NaN	NaN	10.42	4.58	0.00	8.00	11.00	14.00	20.00

[84]:
```
por.describe(include='all').T
```

[84]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
school	649	2	GP	423	NaN	NaN	NaN	NaN	NaN	NaN	NaN
sex	649	2	F	383	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	649.00	NaN	NaN	NaN	16.74	1.22	15.00	16.00	17.00	18.00	22.00
address	649	2	U	452	NaN	NaN	NaN	NaN	NaN	NaN	NaN
famsize	649	2	GT3	457	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Pstatus	649	2	T	569	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Medu	649.00	NaN	NaN	NaN	2.51	1.13	0.00	2.00	2.00	4.00	4.00
Fedu	649.00	NaN	NaN	NaN	2.31	1.10	0.00	1.00	2.00	3.00	4.00
Mjob	649	5	other	258	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Fjob	649	5	other	367	NaN	NaN	NaN	NaN	NaN	NaN	NaN
reason	649	4	course	285	NaN	NaN	NaN	NaN	NaN	NaN	NaN
guardian	649	3	mother	455	NaN	NaN	NaN	NaN	NaN	NaN	NaN
traveltime	649.00	NaN	NaN	NaN	1.57	0.75	1.00	1.00	1.00	2.00	4.00
studytime	649.00	NaN	NaN	NaN	1.93	0.83	1.00	1.00	2.00	2.00	4.00
failures	649.00	NaN	NaN	NaN	0.22	0.59	0.00	0.00	0.00	0.00	3.00
schoolsup	649	2	no	581	NaN	NaN	NaN	NaN	NaN	NaN	NaN
famsup	649	2	yes	398	NaN	NaN	NaN	NaN	NaN	NaN	NaN
paid	649	2	no	610	NaN	NaN	NaN	NaN	NaN	NaN	NaN
activities	649	2	no	334	NaN	NaN	NaN	NaN	NaN	NaN	NaN
nursery	649	2	yes	521	NaN	NaN	NaN	NaN	NaN	NaN	NaN
higher	649	2	yes	580	NaN	NaN	NaN	NaN	NaN	NaN	NaN
internet	649	2	yes	498	NaN	NaN	NaN	NaN	NaN	NaN	NaN
romantic	649	2	no	410	NaN	NaN	NaN	NaN	NaN	NaN	NaN
famrel	649.00	NaN	NaN	NaN	3.93	0.96	1.00	4.00	4.00	5.00	5.00
freetime	649.00	NaN	NaN	NaN	3.18	1.05	1.00	3.00	3.00	4.00	5.00
goout	649.00	NaN	NaN	NaN	3.18	1.18	1.00	2.00	3.00	4.00	5.00
Dalc	649.00	NaN	NaN	NaN	1.50	0.92	1.00	1.00	1.00	2.00	5.00
Walc	649.00	NaN	NaN	NaN	2.28	1.28	1.00	1.00	2.00	3.00	5.00
health	649.00	NaN	NaN	NaN	3.54	1.45	1.00	2.00	4.00	5.00	5.00
absences	649.00	NaN	NaN	NaN	3.66	4.64	0.00	0.00	2.00	6.00	32.00
G3	649.00	NaN	NaN	NaN	11.91	3.23	0.00	10.00	12.00	14.00	19.00

8.1 Organizando o Dataset

Lendo a descrição das variáveis e a descrição estatística feita acima, optei por remover duas do dataset.

Há tanto variáveis sobre o nível de educação quanto o tipo de emprego dos pais. Entendo que há uma diferença entre as duas variáveis, mas, mais de 35% do Mjob e mais de 54% do Fjob são 'other'. Ou seja, é como se 35 e 55% pas variáveis fossem nulas (no sentido de que não trazem um insight sobre o efeito de determinada profissão).

Então, por questões de quantidade de dummies no modelo, manterei somente as variáveis de nível de educação - há um concenso que o nível de educação dos pais afeta a educação dos filhos.

Assim, droparei as variáveis *Mjob e Fjob*

[85]:
```python
mat = mat.drop(['Mjob', 'Fjob'], 1)
por = por.drop(['Mjob', 'Fjob'], 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
```

Começando com a transformação das variáveis binárias em numéricas

[86]:
```
cols = ['schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher',␣
↪'internet', 'romantic']
mat[cols] = mat[cols].replace({'yes': 1, 'no': 0})
por[cols] = por[cols].replace({'yes': 1, 'no': 0})
```

Agora transformando as variáveis 'school', 'address', 'famsize'. 'guardian', 'reason' e 'Pstatus' em dummy e depois dropando uma das colunas dummies gerada para cada variável.

Isso pois não é necessário manter as duas (ou três) já que 0 significa ausência de uma e, portanto, presença da outra.

[87]:
```
mat = pd.get_dummies(
    mat, columns=['sex','school', 'address', 'famsize', 'Pstatus', 'guardian',␣
↪'reason'])
mat = mat.drop(['sex_M','school_MS', 'address_R', 'famsize_GT3', 'Pstatus_T',␣
↪'guardian_other', 'reason_other'], 1)

por = pd.get_dummies(
    por, columns=['sex','school', 'address', 'famsize', 'Pstatus', 'guardian',␣
↪'reason'])
por = por.drop(['sex_M','school_MS', 'address_R', 'famsize_GT3', 'Pstatus_T',␣
↪'guardian_other', 'reason_other'], 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  This is separate from the ipykernel package so we can avoid doing imports
until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  import sys
```

[88]:
```
# Ordenando os dfs para começar com a variável de interesse - G3

cols_to_order = ['G3']
new_columns = cols_to_order + (mat.columns.drop(cols_to_order).tolist())
mat = mat[new_columns]

cols_to_order = ['G3']
new_columns = cols_to_order + (por.columns.drop(cols_to_order).tolist())
por = por[new_columns]
```

[89]:
```
mat.head()
```

[89]:
```
   G3  age  Medu  Fedu  traveltime  studytime  failures  schoolsup  famsup  \
0   6   18     4     4           2          2         0          1       0
1   6   17     1     1           1          2         0          0       1
2  10   15     1     1           1          2         3          1       0
3  15   15     4     2           1          3         0          0       1
4  10   16     3     3           1          2         0          0       1

   paid  …  sex_F  school_GP  address_U  famsize_LE3  Pstatus_A  \
0     0  …      1          1          1            0          1
1     0  …      1          1          1            0          0
2     1  …      1          1          1            1          0
3     1  …      1          1          1            0          0
4     1  …      1          1          1            0          0

   guardian_father  guardian_mother  reason_course  reason_home  \
0                0                1              1            0
1                1                0              1            0
2                0                1              0            0
3                0                1              0            1
4                1                0              0            1

   reason_reputation
0                  0
1                  0
2                  0
3                  0
4                  0

[5 rows x 32 columns]
```

[90]:
```
por.head()
```

[90]:
```
   G3  age  Medu  Fedu  traveltime  studytime  failures  schoolsup  famsup  \
0  11   18     4     4           2          2         0          1       0
1  11   17     1     1           1          2         0          0       1
2  12   15     1     1           1          2         0          1       0
3  14   15     4     2           1          3         0          0       1
4  13   16     3     3           1          2         0          0       1

   paid  …  sex_F  school_GP  address_U  famsize_LE3  Pstatus_A  \
0     0  …      1          1          1            0          1
1     0  …      1          1          1            0          0
2     0  …      1          1          1            1          0
3     0  …      1          1          1            0          0
4     0  …      1          1          1            0          0
```

```
   guardian_father  guardian_mother  reason_course  reason_home  \
0                0                1              1            0
1                1                0              1            0
2                0                1              0            0
3                0                1              0            1
4                1                0              0            1

   reason_reputation
0                  0
1                  0
2                  0
3                  0
4                  0

[5 rows x 32 columns]
```

8.2 Duplicated values

[91]:
```python
print(mat.duplicated().value_counts())
print(por.duplicated().value_counts())
```

```
False    395
dtype: int64
False    649
dtype: int64
```

Não há valores duplicados que precisem ser removidos.

8.3 Missing Values

[92]:
```python
total = mat.isnull().sum().sort_values(ascending=False)
percent = (mat.isnull().sum()/mat.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

[92]:
```
                 Total  Percent
G3                   0     0.00
age                  0     0.00
reason_home          0     0.00
reason_course        0     0.00
guardian_mother      0     0.00
guardian_father      0     0.00
Pstatus_A            0     0.00
famsize_LE3          0     0.00
```

```
address_U            0     0.00
school_GP            0     0.00
sex_F                0     0.00
absences             0     0.00
health               0     0.00
Walc                 0     0.00
Dalc                 0     0.00
goout                0     0.00
freetime             0     0.00
famrel               0     0.00
romantic             0     0.00
internet             0     0.00
```

[93]:
```
total = por.isnull().sum().sort_values(ascending=False)
percent = (por.isnull().sum()/por.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

[93]:
```
                 Total  Percent
G3                   0     0.00
age                  0     0.00
reason_home          0     0.00
reason_course        0     0.00
guardian_mother      0     0.00
guardian_father      0     0.00
Pstatus_A            0     0.00
famsize_LE3          0     0.00
address_U            0     0.00
school_GP            0     0.00
sex_F                0     0.00
absences             0     0.00
health               0     0.00
Walc                 0     0.00
Dalc                 0     0.00
goout                0     0.00
freetime             0     0.00
famrel               0     0.00
romantic             0     0.00
internet             0     0.00
```

Não há misisng values em nenhuma das bases.

8.4 Train test split

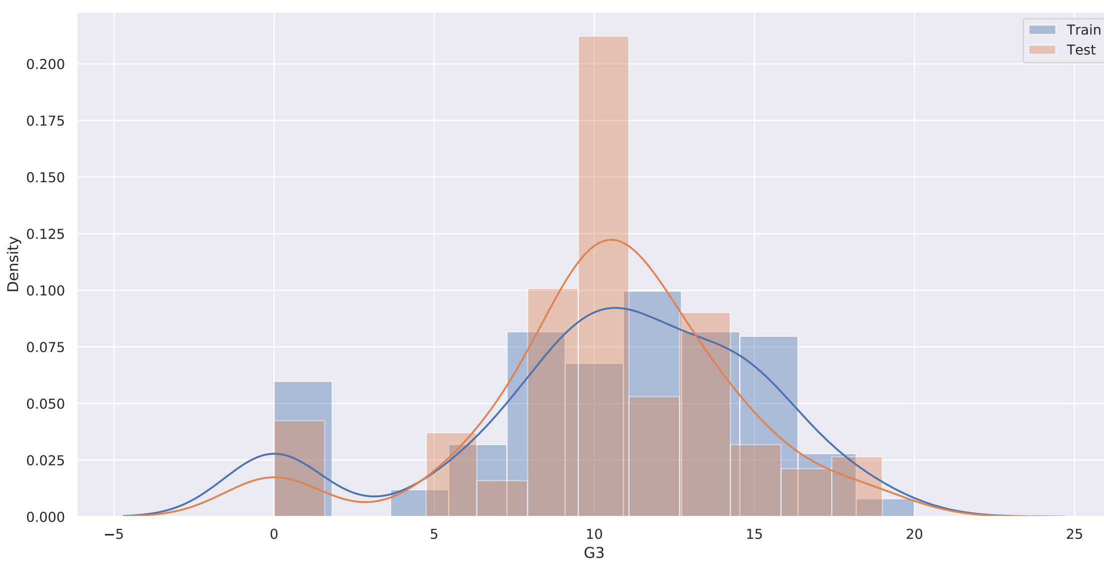
[94]:
```
train_mat, test_mat = train_test_split(mat, test_size=0.3, random_state=1)
```

[95]:
```
train_por, test_por = train_test_split(por, test_size=0.3, random_state=1)
```

8.5 Normalizing the data: Mat

[96]:
```
plt.figure(figsize=(15, 7.5))
sns.distplot(train_mat['G3'], kde = True, label='Train');
sns.distplot(test_mat['G3'], kde = True, label='Test');

plt.legend(loc='upper right')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



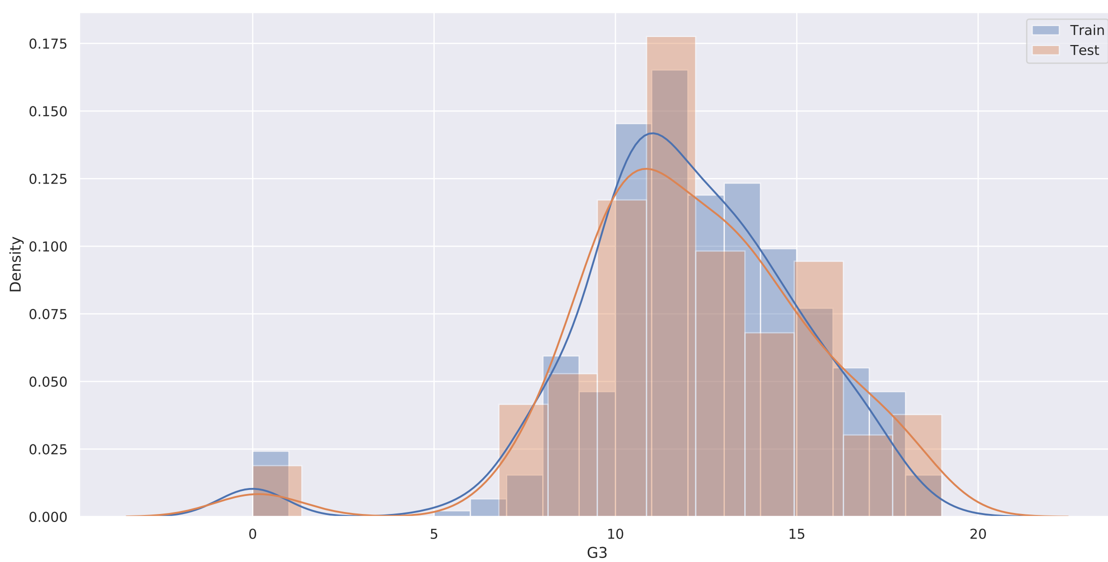
É possível perceber que há uma concentração de observações entre 0 e 2 que 'desnormaliza' a distribuição.

Entretanto, optarei por lidar com isso na parte de outliers, depois de explorar um pouco mais os dados.

8.6 Normalizing the data: Por

[97]:
```
plt.figure(figsize=(15, 7.5))
sns.distplot(train_por['G3'], kde = True, label='Train');
sns.distplot(test_por['G3'], kde = True, label='Test');

plt.legend(loc='upper right')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



8.7 Outlier detection: Mat

Matemática

[98]:
```python
n = 2

upper_limit = train_mat['G3'].mean() + n*train_mat['G3'].std()
lower_limit = train_mat['G3'].mean() - n*train_mat['G3'].std()

print("Highest allowed", upper_limit)
print("Lowest allowed", lower_limit)
```

```
Highest allowed 20.057351599983534
Lowest allowed 0.8122136174077728
```

Áplicando a regra de trimming

[99]:
```python
train_trimmed = train_mat[
    ( train_mat['G3'] < upper_limit ) &

    ( train_mat['G3'] > lower_limit )
    ]
```

[100]:
```python
train_mat.shape
```

[100]: (276, 32)

[101]:
```python
train_trimmed.shape #removeu 32 observações
```

[101]: (246, 32)

Aplicando a regra de censoring

[102]:
```python
train_censored = pd.DataFrame()

train_censored['G3'] =  np.where(
    train_mat['G3'] > upper_limit,
        upper_limit,
        np.where(
            train_mat['G3'] < lower_limit,
            lower_limit,
            train_mat['G3']
            )
        )
```

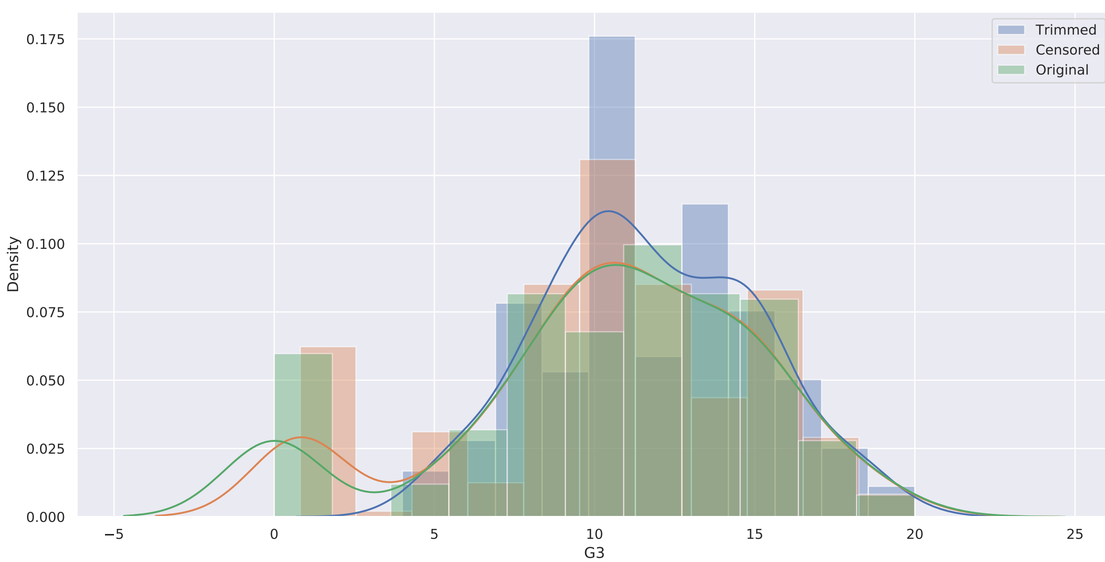
[103]:
```python
train_censored.shape #não removeu nenhuma observação
```

[103]: (276, 1)

Comparando as duas regras

[104]:
```
plt.figure(figsize=(15, 7.5))
sns.distplot(train_trimmed['G3'], kde = True, label='Trimmed');
sns.distplot(train_censored['G3'], kde = True, label='Censored');
sns.distplot(train_mat['G3'], kde = True, label='Original')

plt.legend(loc='upper right')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



A regra escolhida foi a trimmed pois removeu observações e tornou a distribuição mais próxima da normal.

[105]:

```
train_trimmed
```

[105]:

```
     G3  age  Medu  Fedu  traveltime  studytime  failures  schoolsup  famsup  \
348  15   17     4     3           1          3         0          0       1
59   16   16     4     2           1          2         0          0       1
120  15   15     1     2           1          2         0          0       0
12   14   15     4     4           1          1         0          0       1
306  18   20     3     2           1          1         0          0       0
..   ..  …    …   …         …          …         …         …       …
203   6   17     2     2           1          1         0          0       1
255   8   17     1     1           2          1         1          0       1
72    5   15     1     1           1          2         2          1       1
235  10   16     3     2           2          3         0          0       0
37   15   16     4     4           2          3         0          0       1

     paid  …  sex_F  school_GP  address_U  famsize_LE3  Pstatus_A  \
348     1  …      1          1          1            0          0
59      0  …      1          1          1            0          0
120     0  …      1          1          1            0          0
12      1  …      0          1          1            1          0
306     0  …      0          1          1            0          1
..    …  …    …        …        …          …        …
203     0  …      1          1          0            0          0
255     0  …      0          1          1            1          0
72      0  …      1          1          0            0          0
235     0  …      0          1          1            0          0
37      0  …      0          1          0            0          1

     guardian_father  guardian_mother  reason_course  reason_home  \
348                0                1              0            0
59                 0                1              1            0
120                0                1              1            0
12                 1                0              1            0
306                0                0              1            0
..               …              …            …          …
203                0                1              0            0
255                0                1              1            0
72                 0                1              0            0
235                0                1              0            0
37                 0                1              0            0

     reason_reputation
348                  1
59                   0
120                  0
12                   0
306                  0
```

```
..                   …
203                   1
255                   0
72                    1
235                   1
37                    1

[246 rows x 32 columns]
```

8.8 Outlier detection: Por

[106]:
```
n = 2

upper_limit = train_por['G3'].mean() + n*train_por['G3'].std()
lower_limit = train_por['G3'].mean() - n*train_por['G3'].std()

print("Highest allowed", upper_limit)
print("Lowest allowed", lower_limit)
```

```
Highest allowed 18.220928977126817
Lowest allowed 5.466295692476706
```

[107]:
```
train_trimmed = train_por[
    (train_por['G3'] < upper_limit ) &

    (train_por['G3'] > lower_limit )
    ]
```

[108]:
```
train_por.shape
```

[108]: (454, 32)

[109]:
```
train_trimmed.shape #removeu 13 observações
```

[109]: (441, 32)

[110]:
```
train_censored = pd.DataFrame()

train_censored['G3'] =  np.where(
    train_por['G3'] > upper_limit,
        upper_limit,
        np.where(
            train_por['G3'] < lower_limit,
            lower_limit,
            train_por['G3']
            )
```

```
        )
```

[111]:
```
train.shape
```

[111]: (1022, 135)

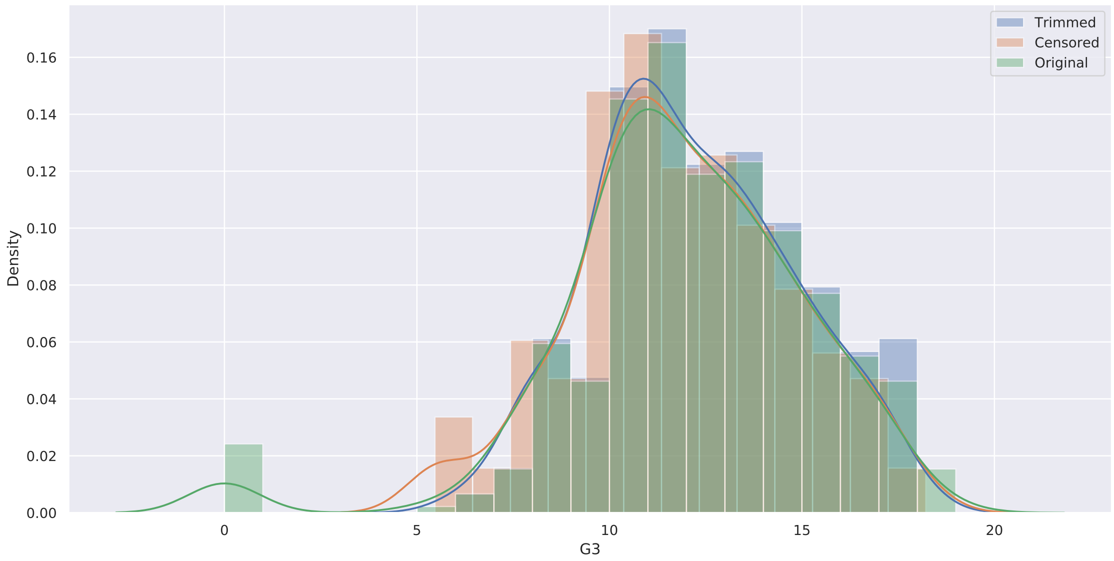
[112]:
```
train_censored.shape #Não excluiu nenhuma observação
```

[112]: (454, 1)

Comparando as duas regras

[113]:
```
plt.figure(figsize=(15, 7.5))
sns.distplot(train_trimmed['G3'], kde = True, label='Trimmed');
sns.distplot(train_censored['G3'], kde = True, label='Censored');
sns.distplot(train_por['G3'], kde = True, label='Original')

plt.legend(loc='upper right')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

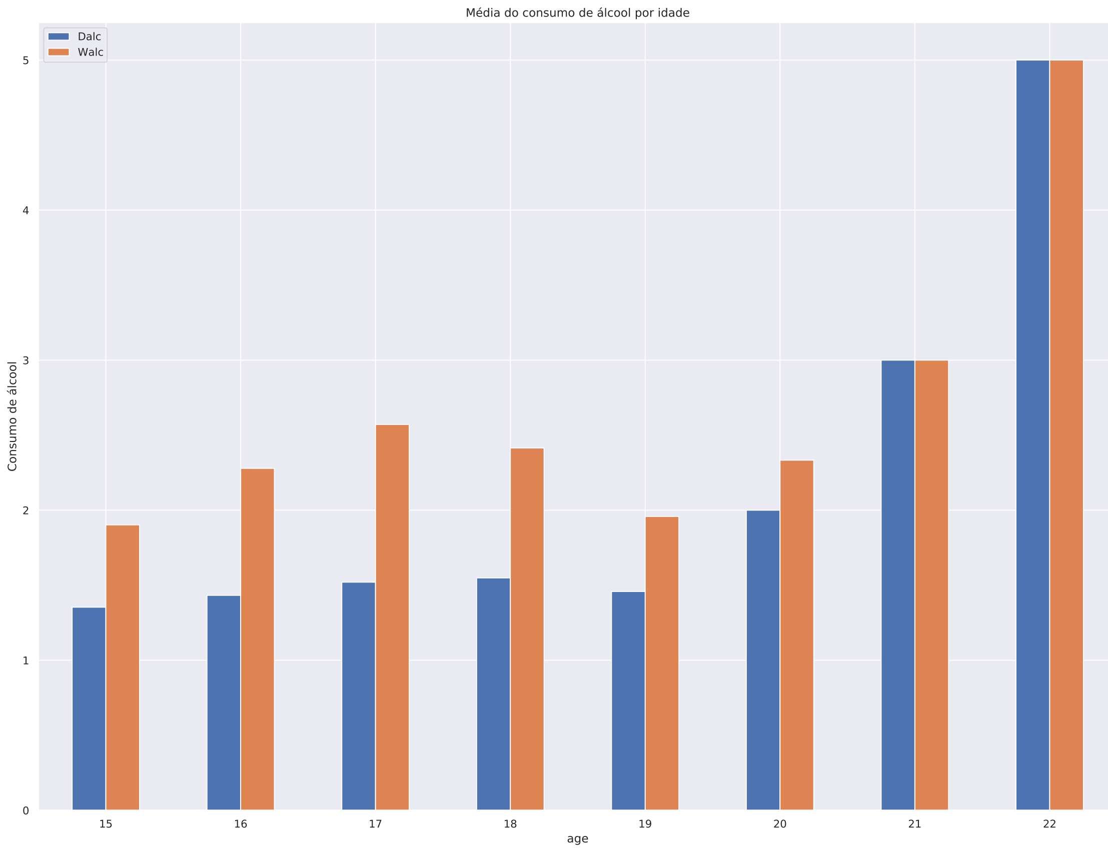


A regra escolhida foi a trimmed pois removeu observações e tornou a distribuição mais próxima da normal.

9 Parte II: Consumo de bebidas

Primeiro, investigarei um pouco como o consumo de álcool está distribuido nos dados.

[114]:
```
# Começando com idade:

mat.groupby('age')[['Dalc', 'Walc']].mean().plot(kind='bar')
plt.ylabel('Consumo de álcool')
plt.xticks(rotation=0)
plt.title('Média do consumo de álcool por idade')
plt.show()
```



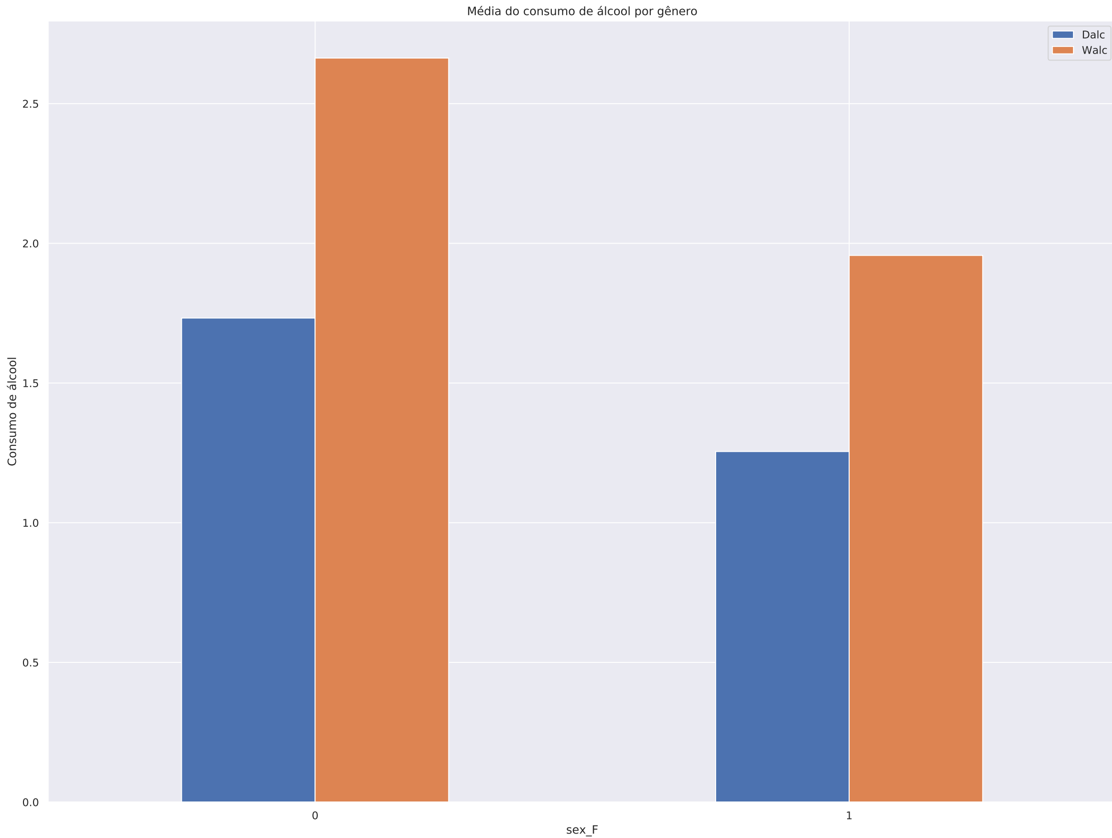
[115]:
```
mat.groupby('age')[['Dalc', 'Walc']].agg(['mean', 'count'])
```

[115]:

	Dalc		Walc	
	mean	count	mean	count
age				
15	1.35	82	1.90	82
16	1.43	104	2.28	104
17	1.52	98	2.57	98
18	1.55	82	2.41	82
19	1.46	24	1.96	24
20	2.00	3	2.33	3
21	3.00	1	3.00	1
22	5.00	1	5.00	1

[116]:
```
# Agora gênero

mat.groupby('sex_F')[['Dalc', 'Walc']].mean().plot(kind='bar')
plt.ylabel('Consumo de álcool')
plt.xticks(rotation=0)
plt.title('Média do consumo de álcool por gênero')
```

```
plt.show()
```

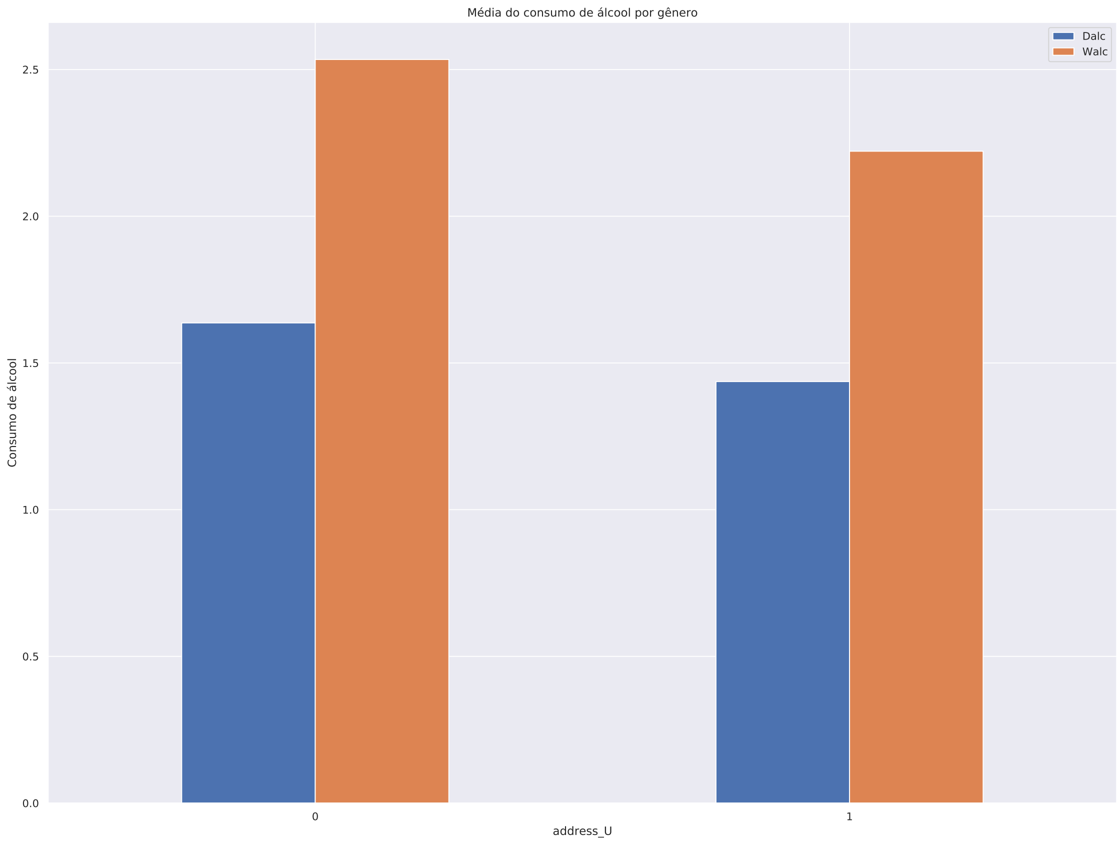


[117]:
```
mat.groupby('sex_F')[['Dalc', 'Walc']].agg(['mean', 'count'])
```

[117]:

	Dalc		Walc	
	mean	count	mean	count
sex_F				
0	1.73	187	2.66	187
1	1.25	208	1.96	208

[118]:
```
# Agora se mora no ambiente urbano ou rural

mat.groupby('address_U')[['Dalc', 'Walc']].mean().plot(kind='bar')
plt.ylabel('Consumo de álcool')
plt.xticks(rotation=0)
plt.title('Média do consumo de álcool por gênero')
plt.show()
```



[119]:
```
mat.groupby('address_U')[['Dalc', 'Walc']].agg(['mean', 'count'])
```

[119]:

	Dalc		Walc	
	mean	count	mean	count
address_U				
0	1.64	88	2.53	88
1	1.44	307	2.22	307

Agora darei uma olhada básica na distribuição de G3 de acordo com consumo de bebidas em dia da semana (Dalc) e em fins de semana (Walc)

[120]:
```
mat.groupby('G3')[['Dalc', 'Walc']].agg(['mean', 'count'])
```

[120]:

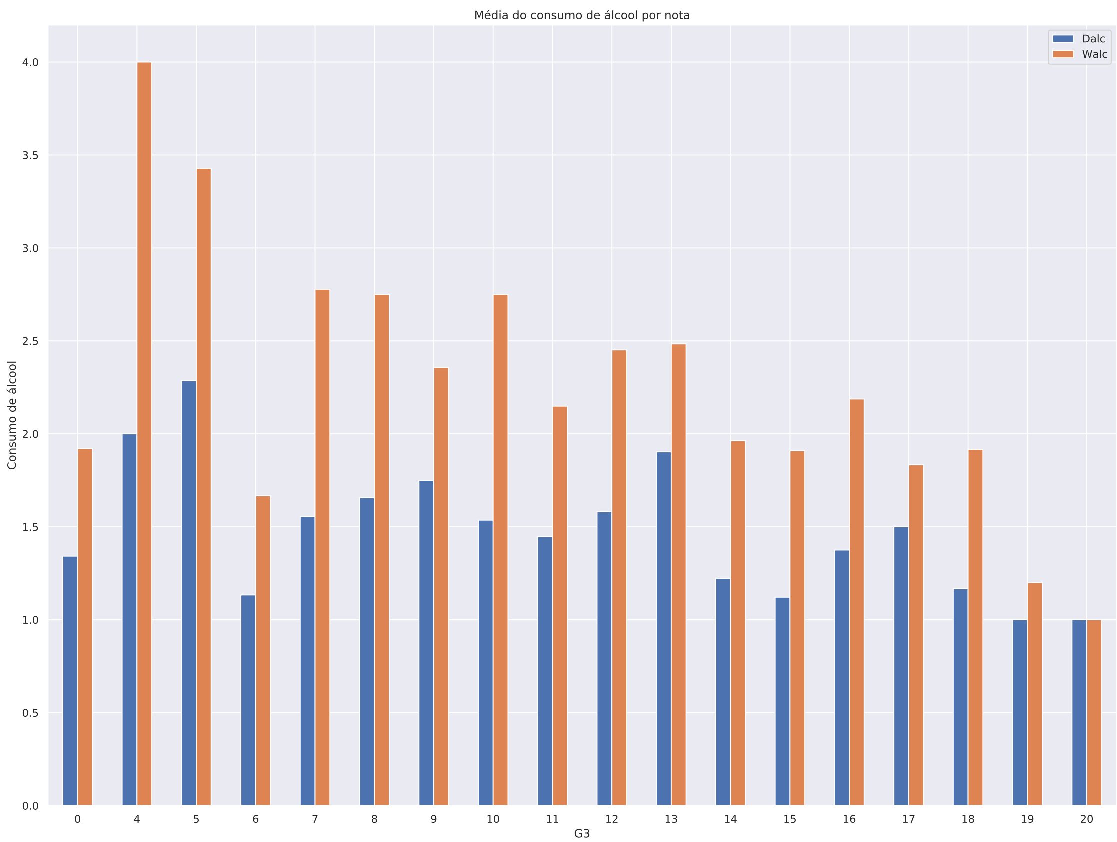
	Dalc		Walc	
	mean	count	mean	count
G3				
0	1.34	38	1.92	38
4	2.00	1	4.00	1
5	2.29	7	3.43	7
6	1.13	15	1.67	15
7	1.56	9	2.78	9

```
8   1.66     32  2.75     32
9   1.75     28  2.36     28
10  1.54     56  2.75     56
11  1.45     47  2.15     47
12  1.58     31  2.45     31
13  1.90     31  2.48     31
14  1.22     27  1.96     27
15  1.12     33  1.91     33
16  1.38     16  2.19     16
17  1.50      6  1.83      6
18  1.17     12  1.92     12
19  1.00      5  1.20      5
20  1.00      1  1.00      1
```

[121]:

```
plt.figure(figsize=(15,6))
mat.groupby('G3')[['Dalc', 'Walc']].mean().plot(kind='bar')
plt.ylabel('Consumo de álcool')
plt.xticks(rotation=0)
plt.title('Média do consumo de álcool por nota')
plt.show()
```

```
<Figure size 1080x432 with 0 Axes>
```



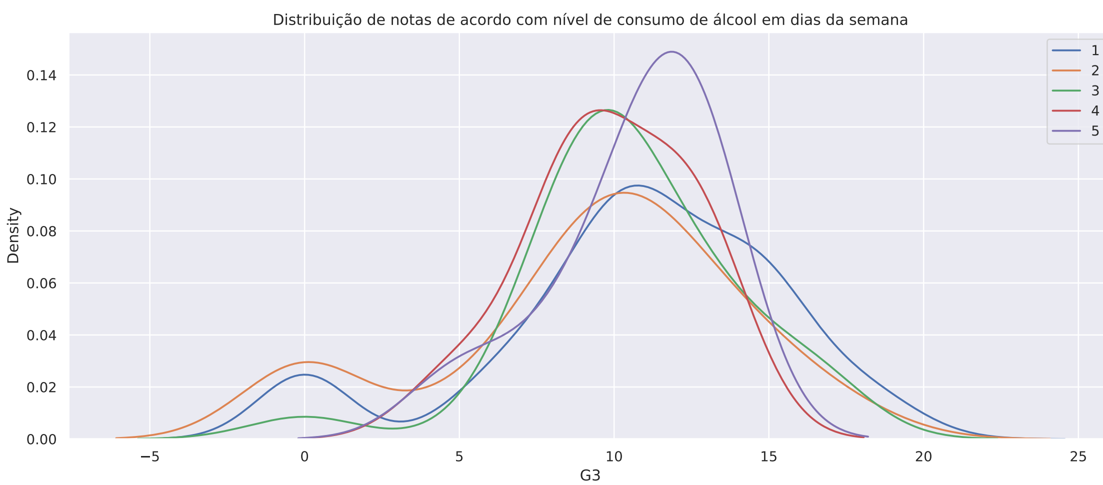
É notável que o consumo de álcool mais alto não é dos alunos com nota 0 e sim dos alunos com notas 4 e 5. Mas, há poucos alunos nessas condições - 8 no total.

[122]:
```
mat.groupby('Dalc')['Dalc'].count().to_frame()

# A maioria dos alunos de matemática se encontram no menor nível de consumo,␣
↪com somente 4.55% estando nos níveis 4 e 5 de consumo.
```

[122]:

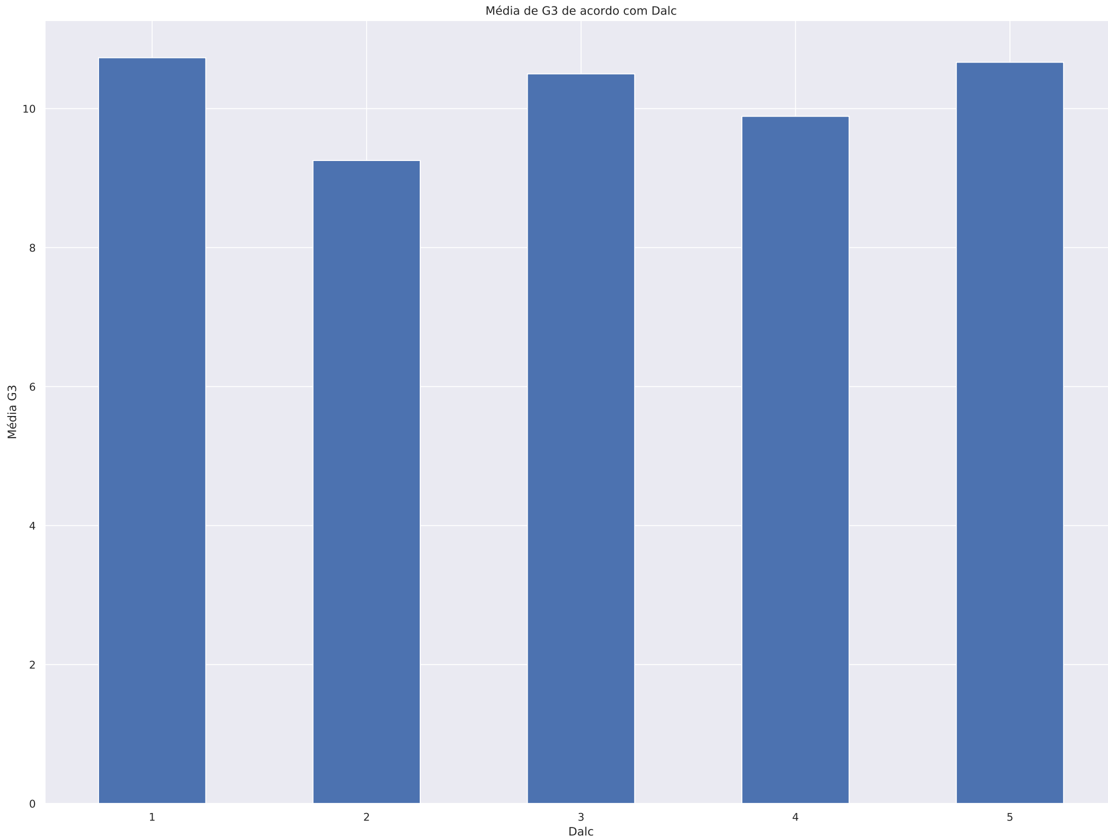
	Dalc
Dalc	
1	276
2	75
3	26
4	9
5	9

[123]:
```
plt.figure(figsize=(15,6))
for dalc, grouped_data in mat.groupby('Dalc'):
    sns.kdeplot(grouped_data['G3'], label=dalc)
plt.legend()
plt.title('Distribuição de notas de acordo com nível de consumo de álcool em␣
↪dias da semana')
plt.show()
```



[124]:
```
mat.groupby('Dalc')['G3'].mean().plot(kind='bar')
plt.title('Média de G3 de acordo com Dalc')
plt.ylabel('Média G3')
plt.xticks(rotation=0)
```

```
plt.show()
```



As distribuições são diferentes: para alunos com consumo baixo (1 e 2) o pico é um pouco mais largo do que para aqueles com consumo alto (5). A média das notas possuem diferenças, mas não necessariamente as imaginadas: as médias dos alunos com consumo 1 e 5 são bem parecidas, com a menor média estando, na verdade, no consumo de nível 2. Entretanto, os grupos tem tamanhos bem diferentes e a amostra não é muito grande, então não é possível ter certeza da relação. Para aprofundas, farei um teste t.

[125]:
```
print(st.ttest_ind(mat.query('Dalc == 1')['G3'], mat.query('Dalc == 2')['G3'],␣
    ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Dalc == 1')['G3'], mat.query('Dalc == 3')['G3'],␣
    ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Dalc == 1')['G3'], mat.query('Dalc == 4')['G3'],␣
    ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Dalc == 1')['G3'], mat.query('Dalc == 5')['G3'],␣
    ↪equal_var=False).pvalue)
```

```
0.0192878681322908
0.7532307681218111
0.3803184594699919
```

0.9461400782666927

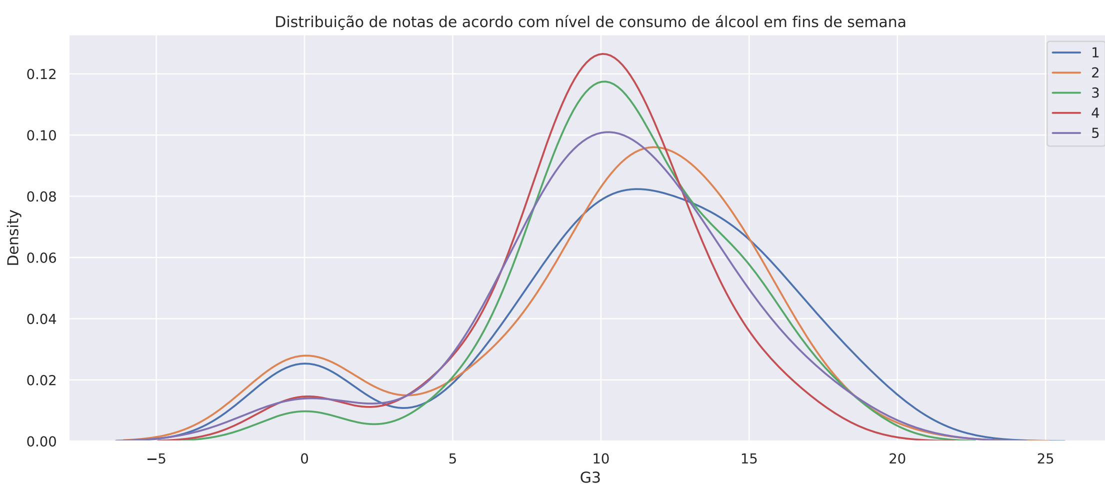
A diferença do grupo 1 para o 2 é significante, enquanto as outras não. Mas, novamente, isso pode ser causado pela pouca quantidade de alunos nos grupos 3, 4 e 5.

[126]:
```
mat.groupby('Walc')['Walc'].count().to_frame()

# A distribuição de Walc é melhor que a de Dalc.
# Apesar da maioria ainda se encontrar no grupo 1, a diferença para os outros␣
↪grupos é um pouco menor.
```

[126]:

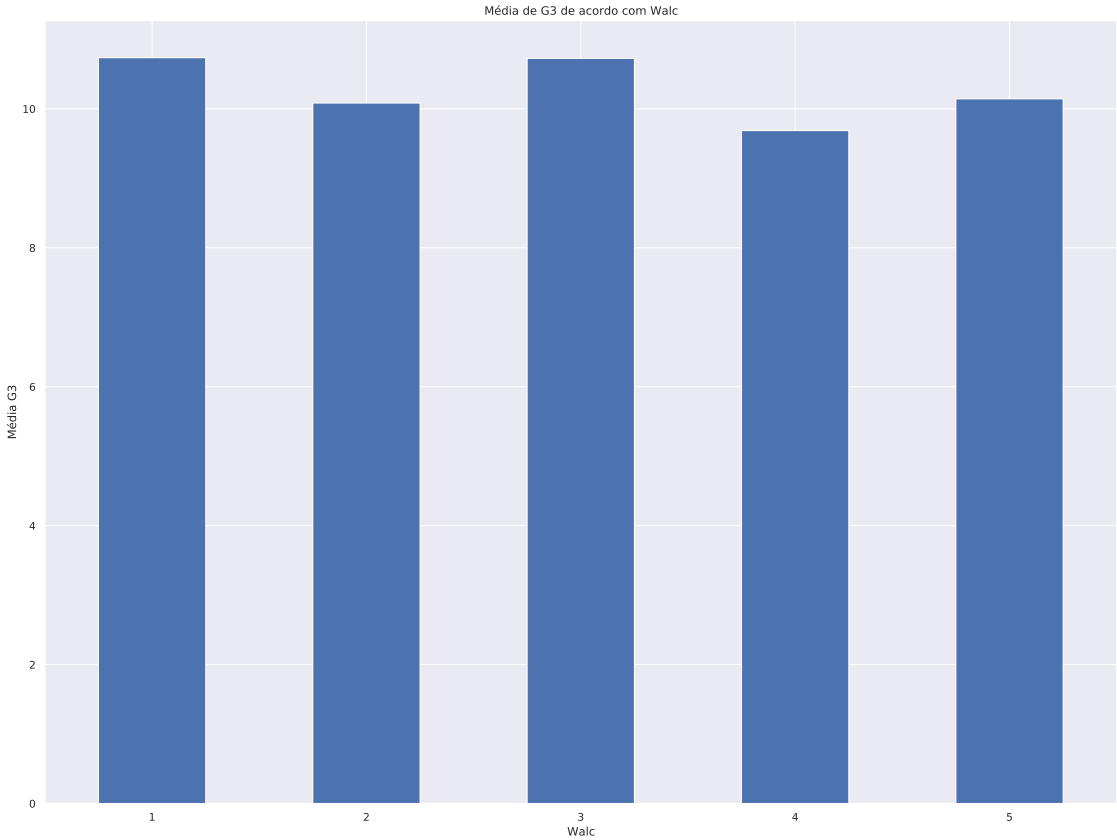
	Walc
Walc	
1	151
2	85
3	80
4	51
5	28

[127]:
```
plt.figure(figsize=(15,6))
for walc, grouped_data in mat.groupby('Walc'):
    sns.kdeplot(grouped_data['G3'], label=walc)
plt.legend()
plt.title('Distribuição de notas de acordo com nível de consumo de álcool em␣
↪fins de semana')
plt.show()
```



[128]:
```
mat.groupby('Walc')['G3'].mean().plot(kind='bar')
plt.title('Média de G3 de acordo com Walc')
plt.ylabel('Média G3')
plt.xticks(rotation=0)
```

```
plt.show()
```



Aqui as distribuições tem picos mais parecidos. A média das notas também tem diferenças pouco gritantes, o que indicaria que não são muito afetadas pelo consumo de álcool

[129]:
```
print(st.ttest_ind(mat.query('Walc == 1')['G3'], mat.query('Walc == 2')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Walc == 1')['G3'], mat.query('Walc == 3')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Walc == 1')['G3'], mat.query('Walc == 4')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Walc == 1')['G3'], mat.query('Walc == 5')['G3'],␣
 ↪equal_var=False).pvalue)
```

```
0.33861317193348284
0.986312737283795
0.11288165231654103
0.5065951077806268
```

Realmente, não há significância estatística na diferença entre os grupos.

9.1 Mas e se agregarmos o consumo de álcool?

É possível que haja um efeito na nota quando consideramos o consumo de álcool total do aluno, o que analisarei aqui.

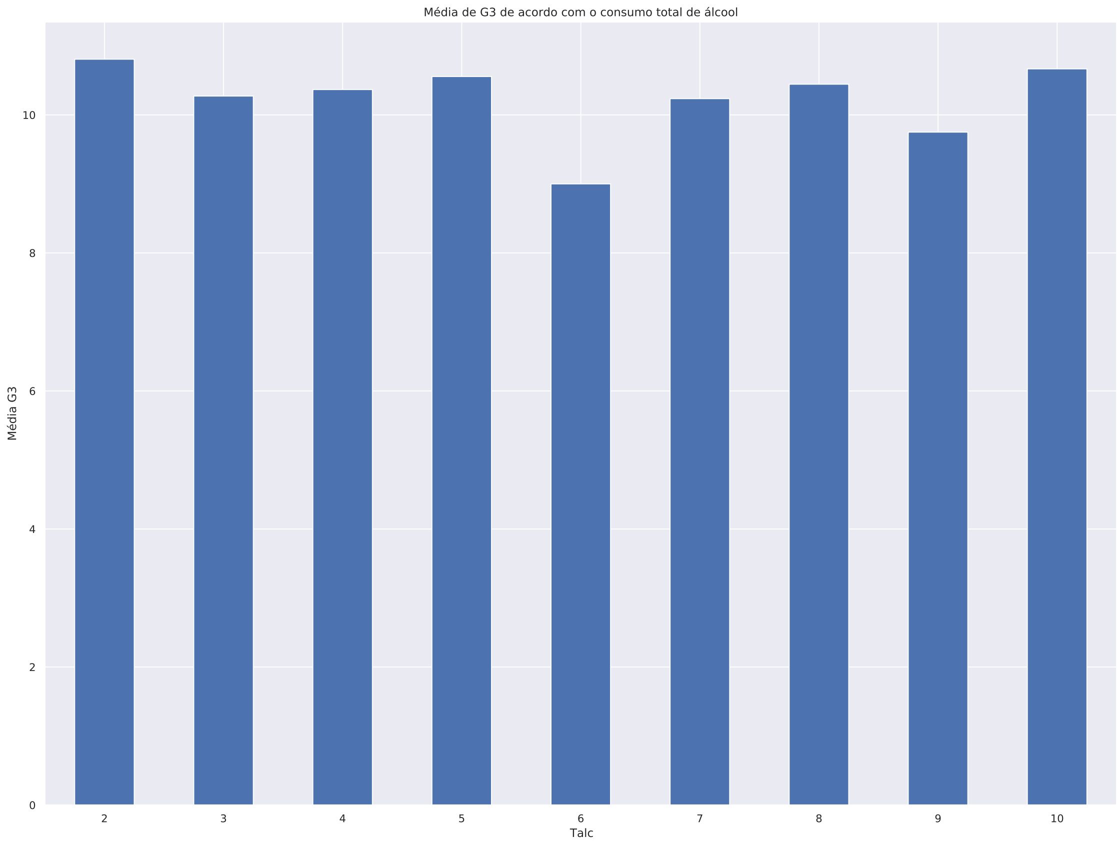
[130]:
```python
mat['Talc'] = mat['Dalc'] + mat['Walc']
```

[131]:
```python
mat.groupby('Talc')['Talc'].count().to_frame()
```

[131]:

Talc	Talc
2	150
3	66
4	60
5	45
6	35
7	17
8	9
9	4
10	9

[132]:
```python
mat.groupby('Talc')['G3'].mean().plot(kind='bar')
plt.title('Média de G3 de acordo com o consumo total de álcool')
plt.ylabel('Média G3')
plt.xticks(rotation=0)
plt.show()
```



[133]:
```
mat.groupby('G3')[['Talc', 'Dalc', 'Walc']].agg(['mean', 'count'])
```

[133]:

	Talc		Dalc		Walc	
	mean	count	mean	count	mean	count
G3						
0	3.26	38	1.34	38	1.92	38
4	6.00	1	2.00	1	4.00	1
5	5.71	7	2.29	7	3.43	7
6	2.80	15	1.13	15	1.67	15
7	4.33	9	1.56	9	2.78	9
8	4.41	32	1.66	32	2.75	32
9	4.11	28	1.75	28	2.36	28
10	4.29	56	1.54	56	2.75	56
11	3.60	47	1.45	47	2.15	47
12	4.03	31	1.58	31	2.45	31
13	4.39	31	1.90	31	2.48	31
14	3.19	27	1.22	27	1.96	27
15	3.03	33	1.12	33	1.91	33
16	3.56	16	1.38	16	2.19	16
17	3.33	6	1.50	6	1.83	6

```
18 3.08     12 1.17     12 1.92     12
19 2.20      5 1.00      5 1.20      5
20 2.00      1 1.00      1 1.00      1
```

[134]:
```python
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 3')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 4')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 5')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 6')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 7')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 8')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 9')['G3'],␣
 ↪equal_var=False).pvalue)
print(st.ttest_ind(mat.query('Talc == 2')['G3'], mat.query('Talc == 10')['G3'],␣
 ↪equal_var=False).pvalue)
```

```
0.4730833845032465
0.5265773086822543
0.7089327119651319
0.024576900806196168
0.6090765367765631
0.8304620937577545
0.42370177738369236
0.8897801960595911
```

Somente o nível 6 de consumo total aparenta ter algum efeito estatisticamente siginificante.

[135]:
```python
mat = mat.drop(['Talc'], 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

9.2 Regressões só com consumo de álcool

[136]:
```python
train_alc, test_alc = train_test_split(mat, test_size=0.3, random_state=7)

X_lr = sm.add_constant( train_alc[['Dalc', 'Walc']] )
X_lr_test = sm.add_constant( test_alc[['Dalc', 'Walc']])
linear_reg = LinearRegression()
```

```
linear_reg = sm.OLS(train_alc[['G3']], X_lr  )
linear_reg_fit  = linear_reg.fit()
linear_reg_pred = linear_reg_fit.predict(X_lr_test)
print(linear_reg_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     G3   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                 -0.003
Method:                 Least Squares   F-statistic:                    0.5637
Date:                Sun, 15 May 2022   Prob (F-statistic):              0.570
Time:                        18:57:49   Log-Likelihood:                -818.63
No. Observations:                 276   AIC:                             1643.
Df Residuals:                     273   BIC:                             1654.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         10.5717      0.610     17.332      0.000       9.371      11.773
Dalc           0.0263      0.417      0.063      0.950      -0.795       0.848
Walc          -0.2470      0.293     -0.844      0.399      -0.823       0.329
==============================================================================
Omnibus:                       20.805   Durbin-Watson:                   2.097
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               23.486
Skew:                          -0.707   Prob(JB):                     7.94e-06
Kurtosis:                       3.208   Cond. No.                         7.21
==============================================================================

Warnings:
[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:117:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[137]:
```
# Ridge

ridge_reg = Ridge(alpha = 0.5)
ridge_reg.fit(train_alc[['Dalc', 'Walc']], train_alc[['G3']] )
ridge_pred = ridge_reg.predict(test_alc[['Dalc', 'Walc']])
```

[138]:
```
ridge_reg.coef_
```

[138]:
```
array([[ 0.02576615, -0.24649678]])
```

[139]:
```python
# Lasso

model_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(train_alc[['Dalc',␣
↪'Walc']], train_alc[['G3']])

lasso_pred = model_lasso.predict(test_alc[['Dalc', 'Walc']])
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:1571:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[140]:
```python
model_lasso.coef_
```

[140]:
```
array([-0., -0.])
```

[141]:
```python
# Elastic Net

model_ElasticNet = ElasticNetCV(
    l1_ratio = 0.5,
    alphas = [1, 0.1, 0.001, 0.0005],
    fit_intercept = True
).fit(train_alc[['Dalc', 'Walc']], train_alc[['G3']] )

elastic_pred = model_ElasticNet.predict(test_alc[['Dalc', 'Walc']])
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/linear_model/_coordinate_descent.py:1571:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[142]:
```python
model_ElasticNet.coef_
```

[142]:
```
array([-0., -0.])
```

Em conclusão, o consumo de álcool, seja na semana ou em fins de semana, não possui um efeito quantitativo forte nem estatisticamente significante na nota dos alunos.

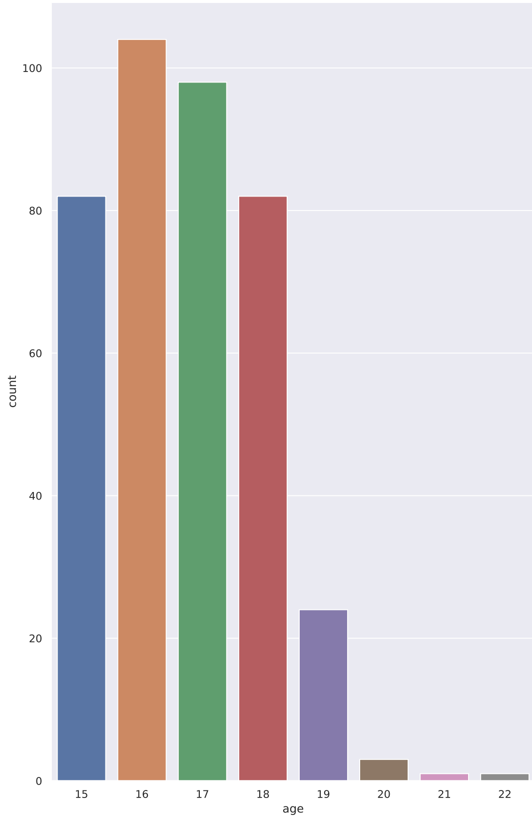
10 Parte III: importância das variáveis

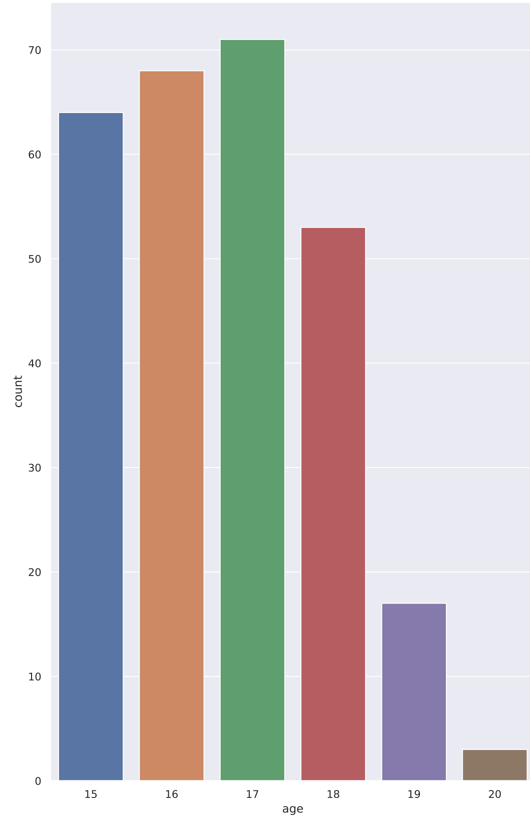
10.1 Matemática

[143]:
```
# Primeiro irei olhar a distribuição das idades e considerar seu efeito em G3.

plt.figure(figsize=(15,6))
fig, ax =plt.subplots(1,2)
sns.countplot(mat['age'], ax=ax[0])
sns.countplot(train_mat['age'], ax=ax[1])
fig.show()

#Aqui fica claro que o conjunto total possui alguns estudantes de 21 e 22 anos,␣
↪enquanto o conjunto de treino possui estudantes até 20 anos.
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. 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
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. 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
```

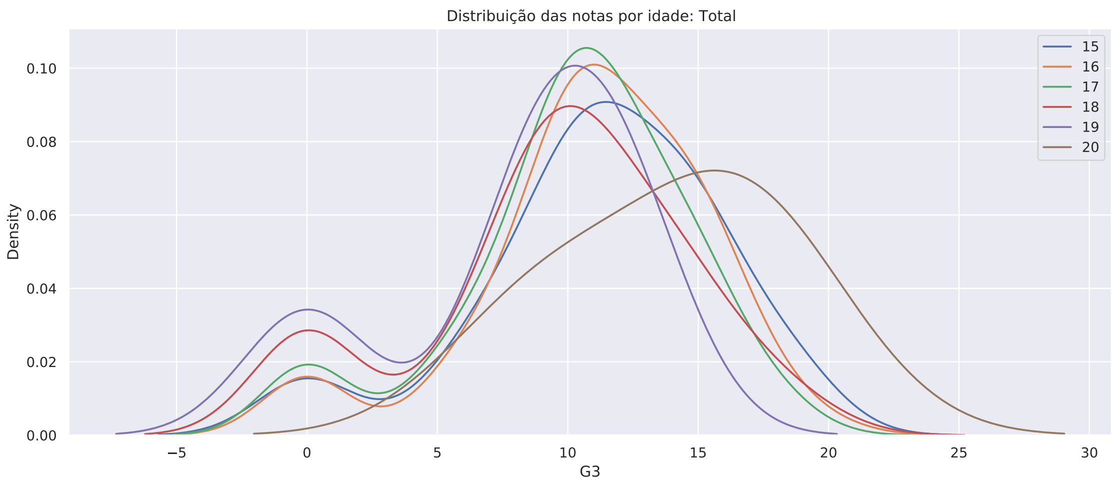
```
<Figure size 1080x432 with 0 Axes>
```



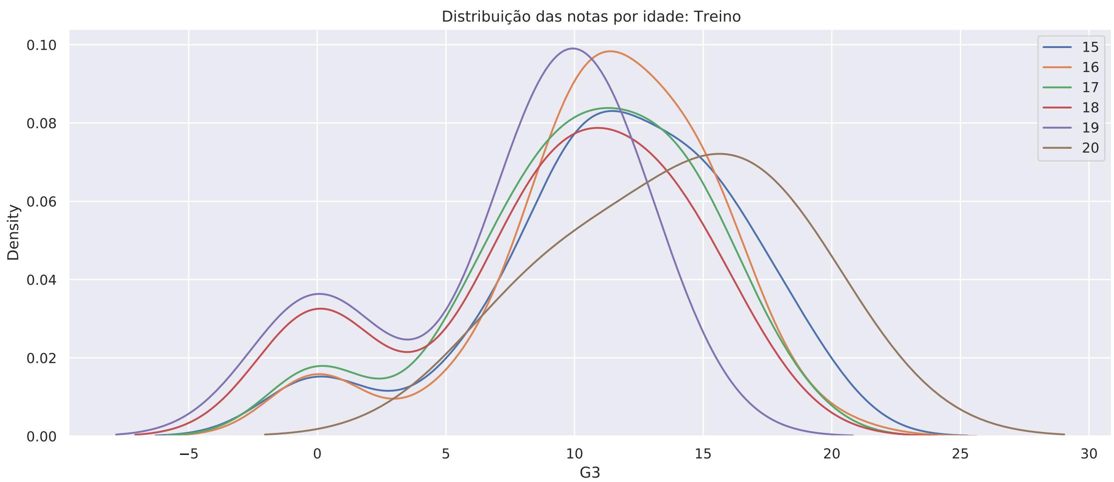


[144]:
```
plt.figure(figsize=(15,6))
for age, grouped_data in mat.groupby('age'):
        sns.kdeplot(grouped_data['G3'], label=age)
plt.legend()
plt.title('Distribuição das notas por idade: Total')
plt.show()

# A distribuição aparenta ser normal para até 19 anos, com 20 puxando um pouco␣
 ↪mais para a direita.
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:316:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:316:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  warnings.warn(msg, UserWarning)
```

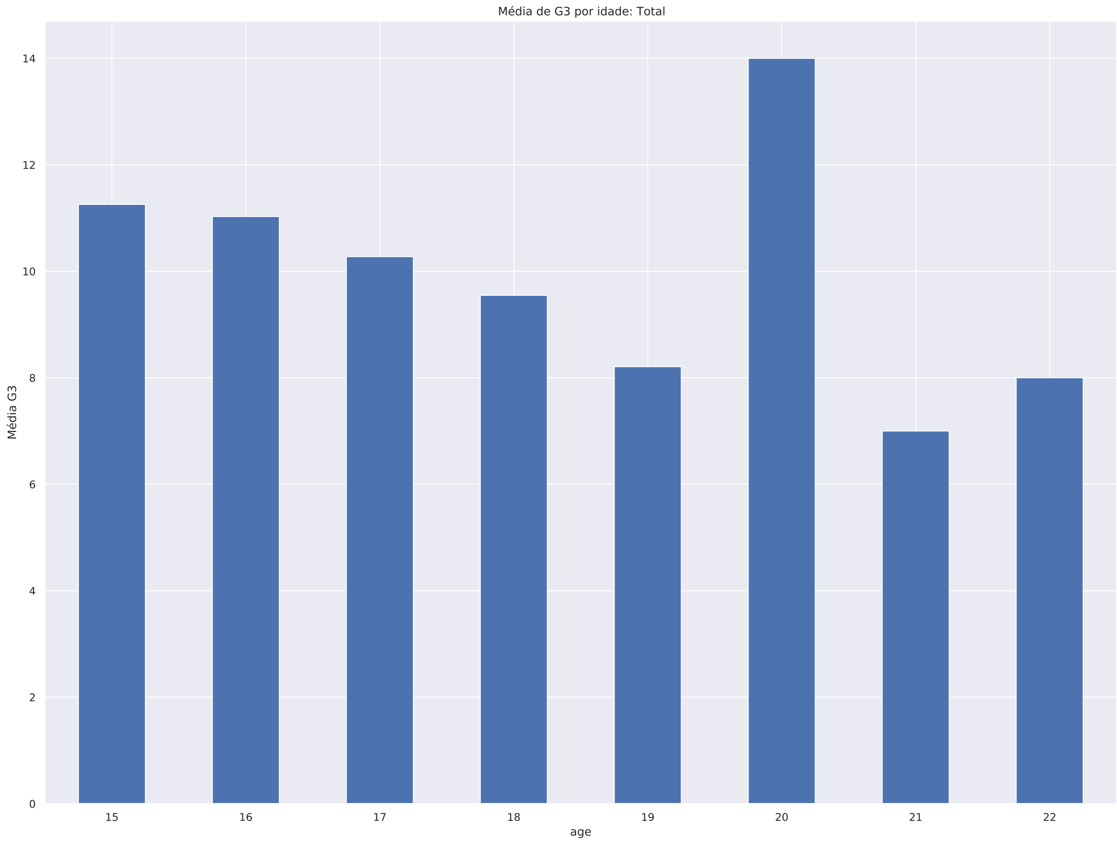


[145]:
```
plt.figure(figsize=(15,6))
for age, grouped_data in train_mat.groupby('age'):
        sns.kdeplot(grouped_data['G3'], label=age)
plt.legend()
plt.title('Distribuição das notas por idade: Treino')
plt.show()

# A distribuição das notas por idade é  similar no conjunto total e no de␣
 ↪treino, com a diferença que o conjunto de treino é um pouco 'puxado' para a␣
 ↪direita.
```

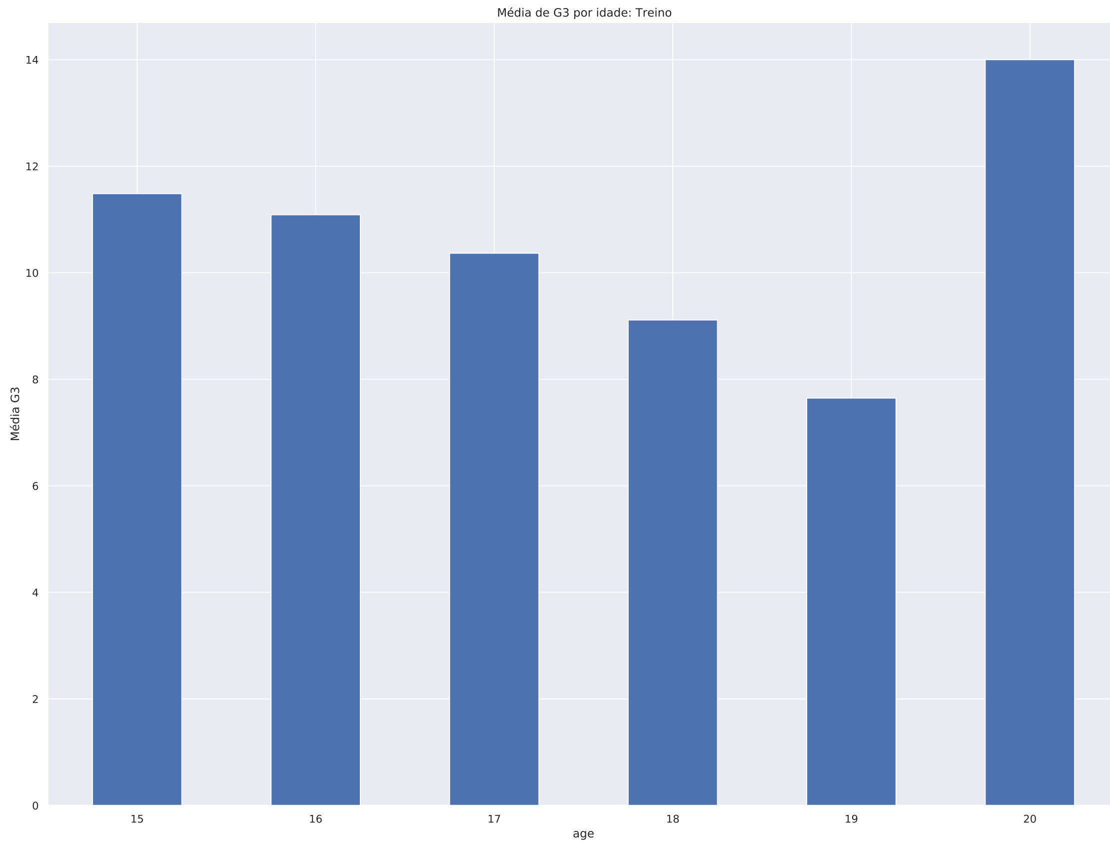


[146]:
```
mat.groupby('age')['G3'].mean().plot(kind='bar')
plt.title('Média de G3 por idade: Total')
plt.ylabel('Média G3')
```

```
plt.xticks(rotation=0)
plt.show()

# É notável que há uma tendência de queda nas notas até os 19 anos. Nos 20,
↪entretanto, os alunos aparentam ter uma melhora considerável na nota - com a
↪média batendo 14.
```



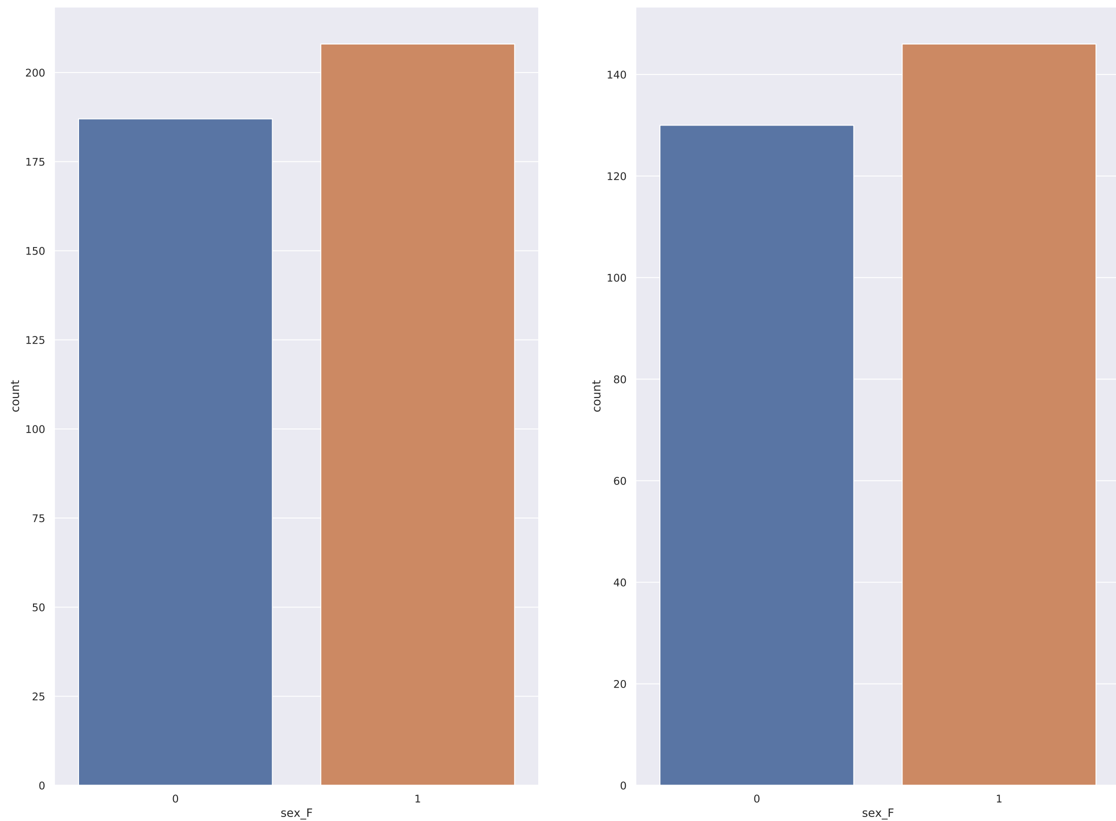
[147]:
```
train_mat.groupby('age')['G3'].mean().plot(kind='bar')
plt.title('Média de G3 por idade: Treino')
plt.ylabel('Média G3')
plt.xticks(rotation=0)
plt.show()

# O mesmo padrão se repete no conjunto de treino:  até os 19 anos há uma
↪tendência de queda e, aos 20, a média sobe.
```



[148]:
```
# Agora vou fazer o mesmo para gênero.

plt.figure(figsize=(15,6))
fig, ax =plt.subplots(1,2)
sns.countplot(mat['sex_F'], ax=ax[0])
sns.countplot(train_mat['sex_F'], ax=ax[1])
fig.show()

# É possível perceber que em ambos os conjuntos há mais mulheres (1) do que␣
↪homens (0)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. 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
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. 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
```

<Figure size 1080x432 with 0 Axes>



[149]:
```
mat.groupby('sex_F')[['G3']].mean()

# Aqui notamos que a média das notas para homens (0) é um pouco maior do que a␣
↪de mulheres (1)
```

[149]:
```
        G3
sex_F
0    10.91
1     9.97
```

[150]:
```
train_mat.groupby('sex_F')[['G3']].mean()

# O mesmo padrão se repete aqui no conjunto de teste
```

[150]:
```
        G3
sex_F
0    10.95
1     9.97
```

Agora irei tentar analisar a importância de cada feature. Novamente, o modelo não está sendo construído aqui, é somente um exercício de observar a importância de cada variável.

Correlação não é um bom modo de definir a importância de cada variável aqui pois temos muitas variáveis dummy e categóricas. Correlação seria uma boa abordagem para variáveis numéricas (que aqui são somente age, failures e absences) Então, irei colocar todas as variáveis numa mesma escala, para que seja possível estimar o peso de cada coeficiente.

[151]:
```
features_t_imp = train_mat.copy().drop(['G3'], axis=1)
target_t_imp = train_mat.copy()['G3']
```

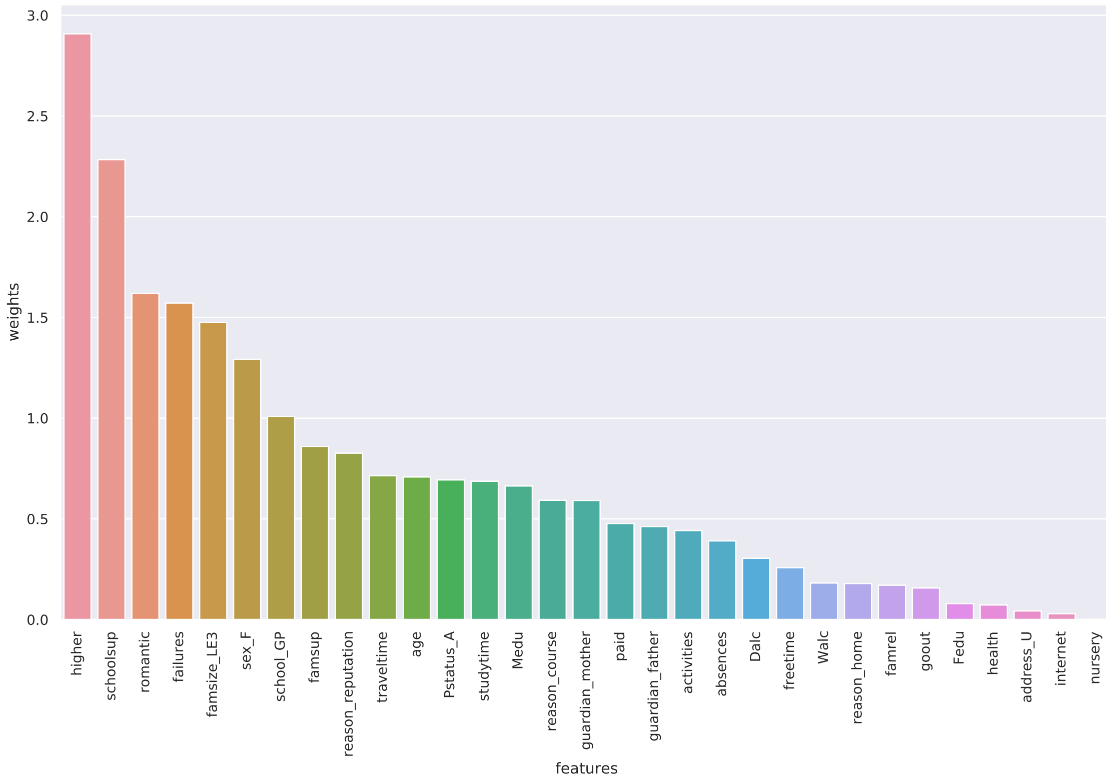
[152]:
```
# Agora é necessário deixar todas as variáveis na mesma escala
# Peguei este método no Kaggle (https://www.kaggle.com/code/ruslansikhamov)

scaler_num = StandardScaler()
features_t_imp[['age', 'absences']] = scaler_num.
 ↪fit_transform(features_t_imp[['age', 'absences']])
```

[153]:
```
linear_regressor = LinearRegression()
linear_regressor.fit(features_t_imp, target_t_imp)
```

[153]: LinearRegression()

[154]:
```
linear_regressor = LinearRegression()
linear_regressor.fit(features_t_imp, target_t_imp)

feature_importances_lr_coef = pd.concat([pd.Series(features_t_imp.columns,␣
 ↪name='features'),
                                         pd.Series(linear_regressor.coef_,␣
 ↪name='weights')],
                                        axis=1) #criando df com os pesos das␣
 ↪features

feature_importances_lr_coef['weights'] =␣
 ↪abs(feature_importances_lr_coef['weights']) # como quero saber o peso, vou␣
 ↪analisar o valor absoluto
feature_importances_lr_coef = feature_importances_lr_coef.
 ↪sort_values(by='weights', ascending=False).reset_index(drop=True)␣
 ↪#classificando pelo peso
```

[155]:
```
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_lr_coef, x='features', y='weights')
plt.xticks(rotation=90)
plt.show()
```

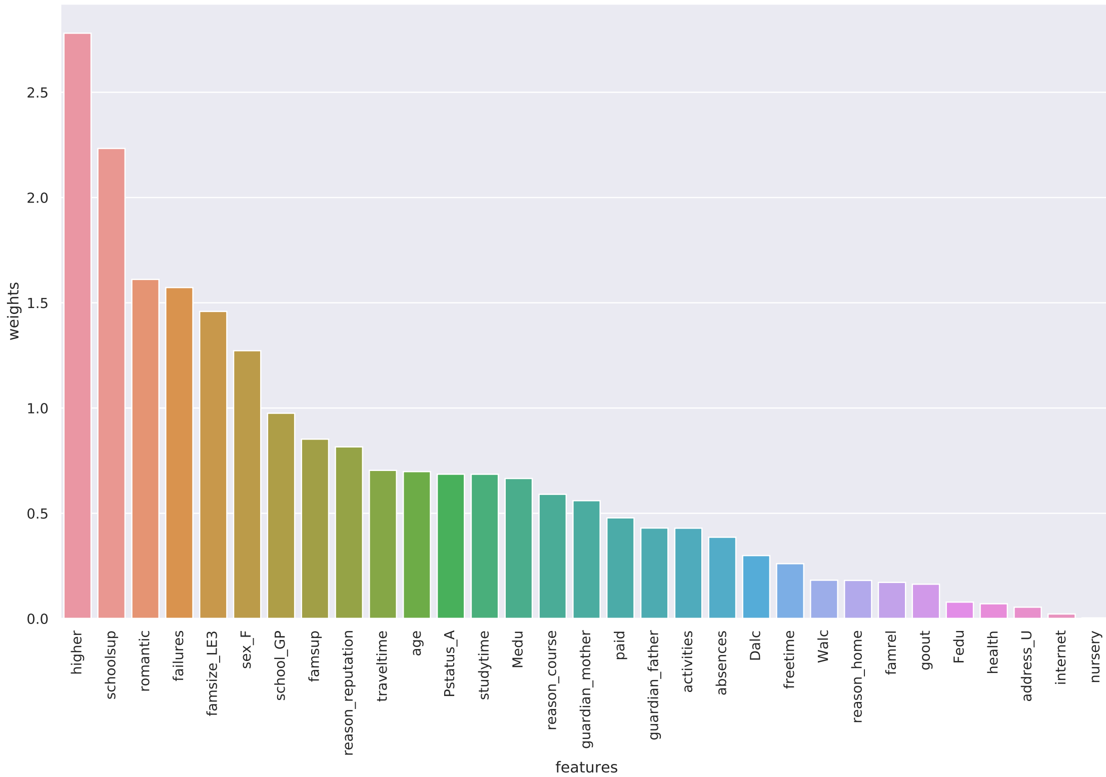


É curioso que, no conjunto de treino, há uma mudança no peso das variáveis. * Querer educação superior passa a ser a variável de maior peso * Suplemento escolar passa a ser a segunda * Em terceiro temos se está em um relacionamento ou não * Failures passados * E ser de uma família LE3

E se usarmos outro tipo de regressão?

[156]:
```
# Ridge

ridge_reg = Ridge(alpha = 0.5)
ridge_reg.fit(features_t_imp, target_t_imp)

feature_importances_rr_coef = pd.concat([pd.Series(features_t_imp.columns,␣
 ↪name='features'),
                                          pd.Series(ridge_reg.coef_,␣
 ↪name='weights')],
                                         axis=1)

feature_importances_rr_coef['weights'] =␣
 ↪abs(feature_importances_rr_coef['weights'])
feature_importances_rr_coef = feature_importances_rr_coef.
 ↪sort_values(by='weights', ascending=False).reset_index(drop=True)
```

[157]:
```
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_rr_coef, x='features', y='weights')
plt.xticks(rotation=90)
plt.show()
```

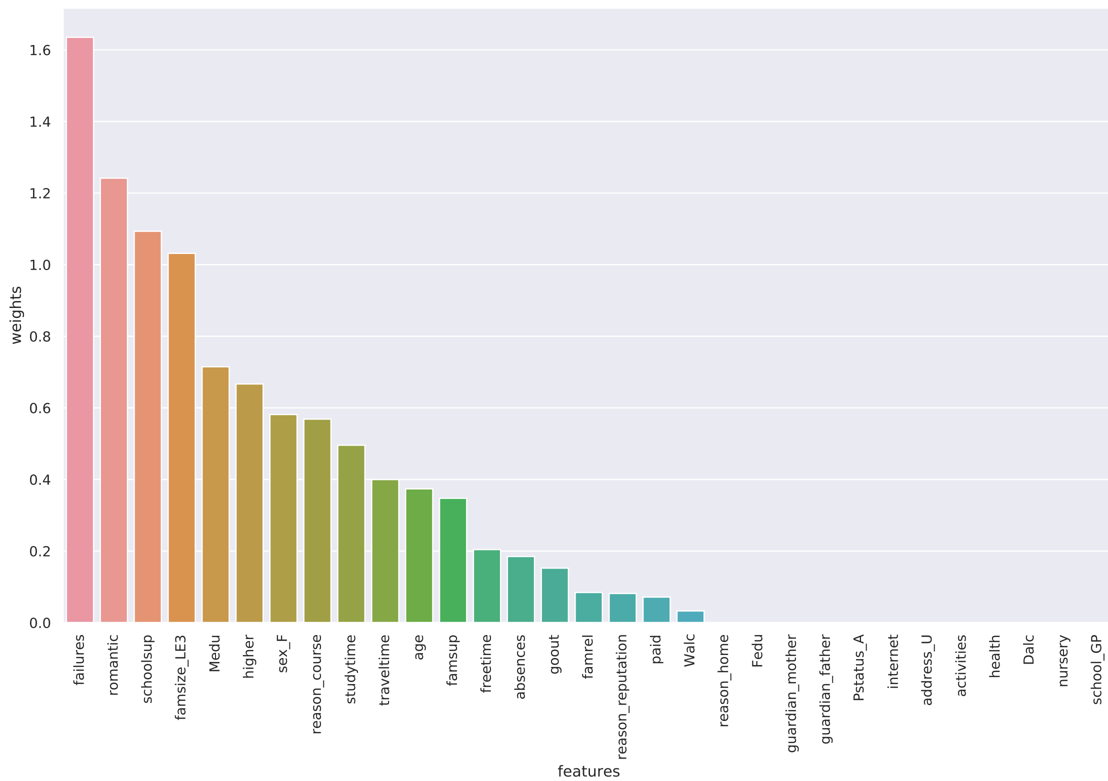


Aqui é notável que as top 5 variáveis são as mesmas do modelo de regressão linear. A diferença na ordem começa a partir da décima variável. Ademais, é possível notar que as variáveis mais para a direita tem peso menor aqui do que no modelo de reg linear.

[158]:
```
# Lasso

model_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(features_t_imp,␣
↪target_t_imp)


feature_importances_lr_coef = pd.concat([pd.Series(features_t_imp.columns,␣
↪name='features'),
                                         pd.Series(model_lasso.coef_,␣
↪name='weights')],
                                        axis=1)
```

```python
feature_importances_lr_coef['weights'] = abs(feature_importances_lr_coef['weights'])
feature_importances_lr_coef = feature_importances_lr_coef.sort_values(by='weights', ascending=False).reset_index(drop=True)
```

[159]:
```python
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_lr_coef, x='features', y='weights')
plt.xticks(rotation=90)
plt.show()
```



Aqui, as top 5 features são:

- failures passados
- se está em um relacionamento
- suplemento escolar
- tamanho da família LE3
- educação da mãe

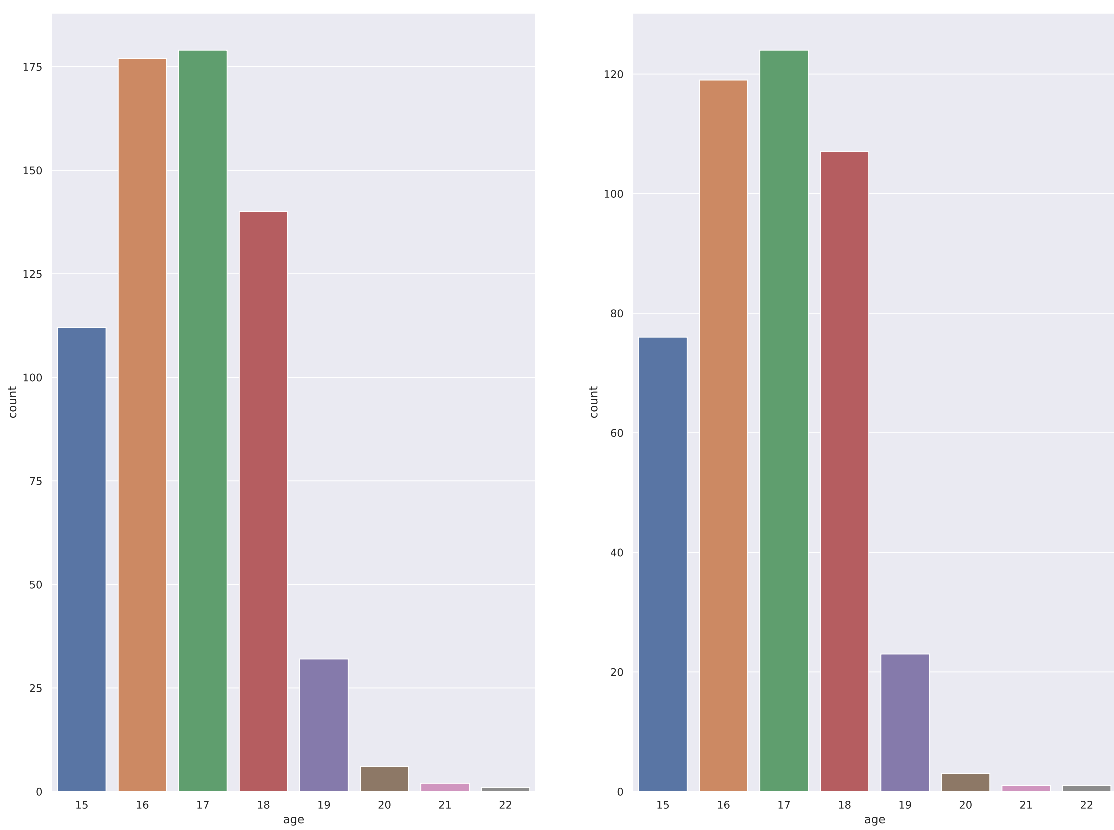
Também é notável que aqui 12 variáveis chegam a ser zeradas, o que não ocorre nem em regressões lineares nem em ridge.

10.2 Português

[160]:
```
plt.figure(figsize=(15,6))
fig, ax =plt.subplots(1,2)
sns.countplot(por['age'], ax=ax[0])
sns.countplot(train_por['age'], ax=ax[1])
fig.show()

# Aqui, diferentemente de mat, há representantes de todas as idades no conjunto␣
↪de treino.
```

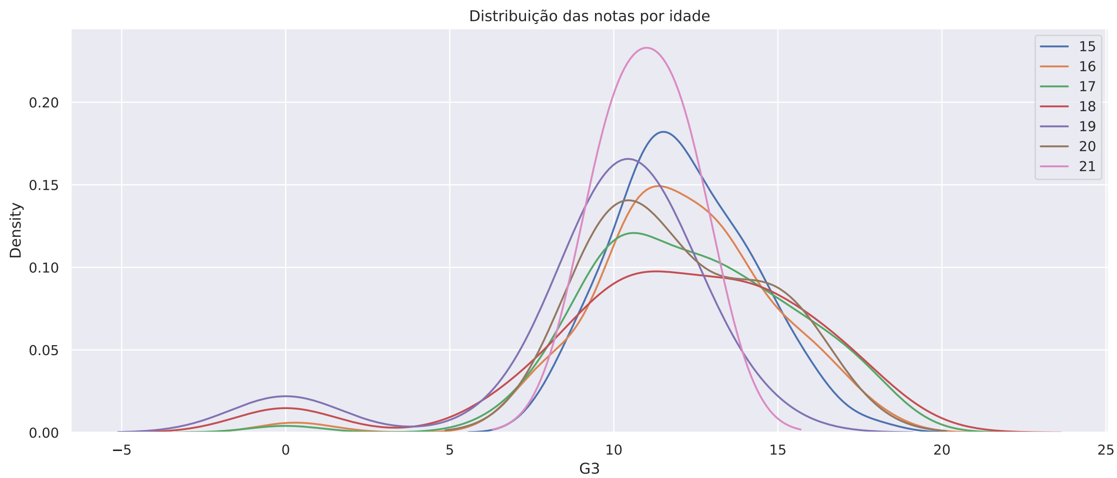
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.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

```
<Figure size 1080x432 with 0 Axes>
```



[161]:
```
plt.figure(figsize=(15,6))
for age, grouped_data in por.groupby('age'):
        sns.kdeplot(grouped_data['G3'], label=age)
plt.legend()
plt.title('Distribuição das notas por idade')
plt.show()
```

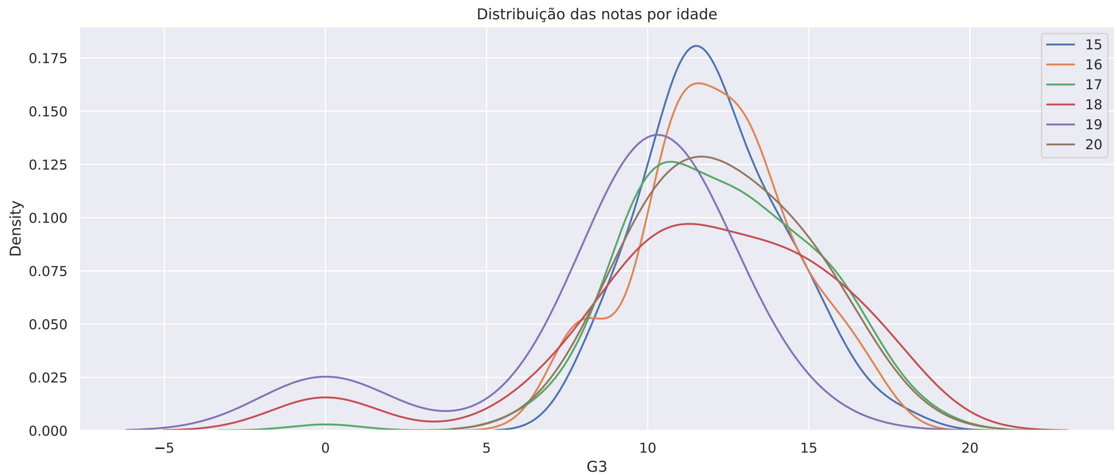
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:316:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  warnings.warn(msg, UserWarning)
```



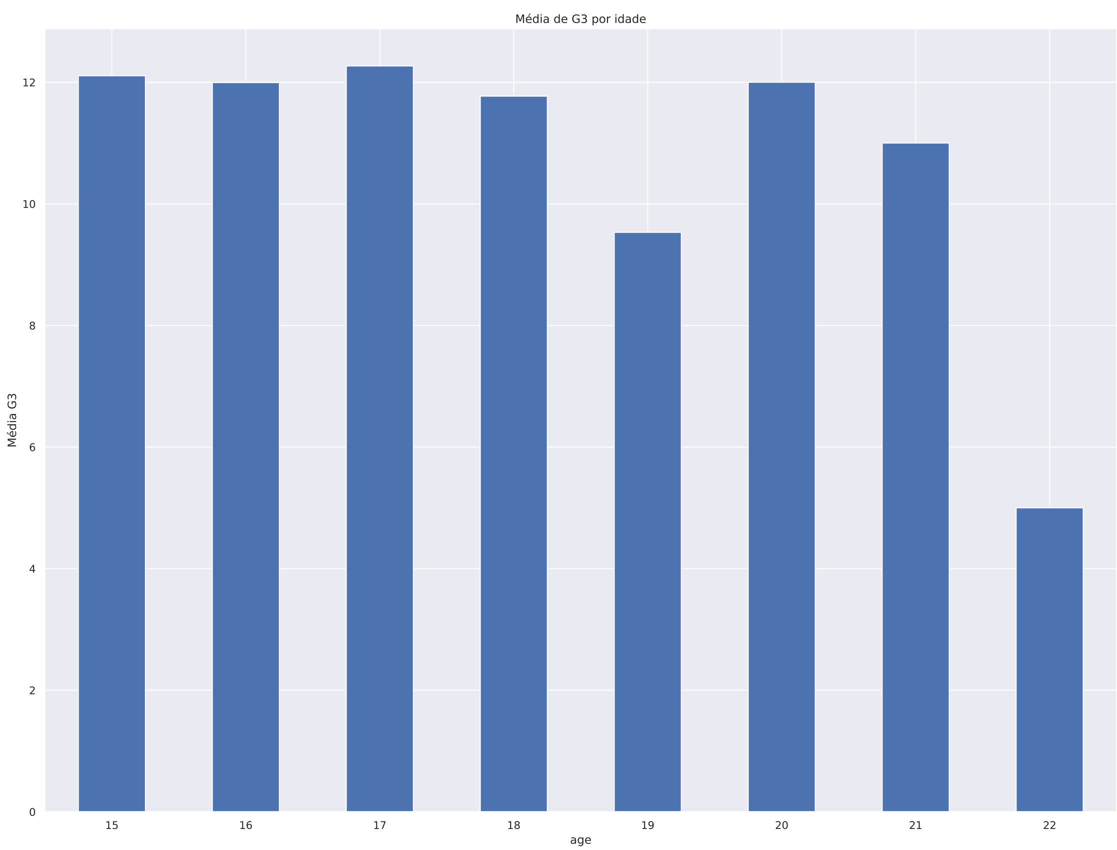
[162]:
```
plt.figure(figsize=(15,6))
for age, grouped_data in train_por.groupby('age'):
        sns.kdeplot(grouped_data['G3'], label=age)
plt.legend()
plt.title('Distribuição das notas por idade')
plt.show()

# Aqui as distribuições não parecem tão 'normais' quanto estavam em mat. Mas␣
 ↪não deixam de estar relativamente similares.
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:316:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:316:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
```

```
warnings.warn(msg, UserWarning)
```

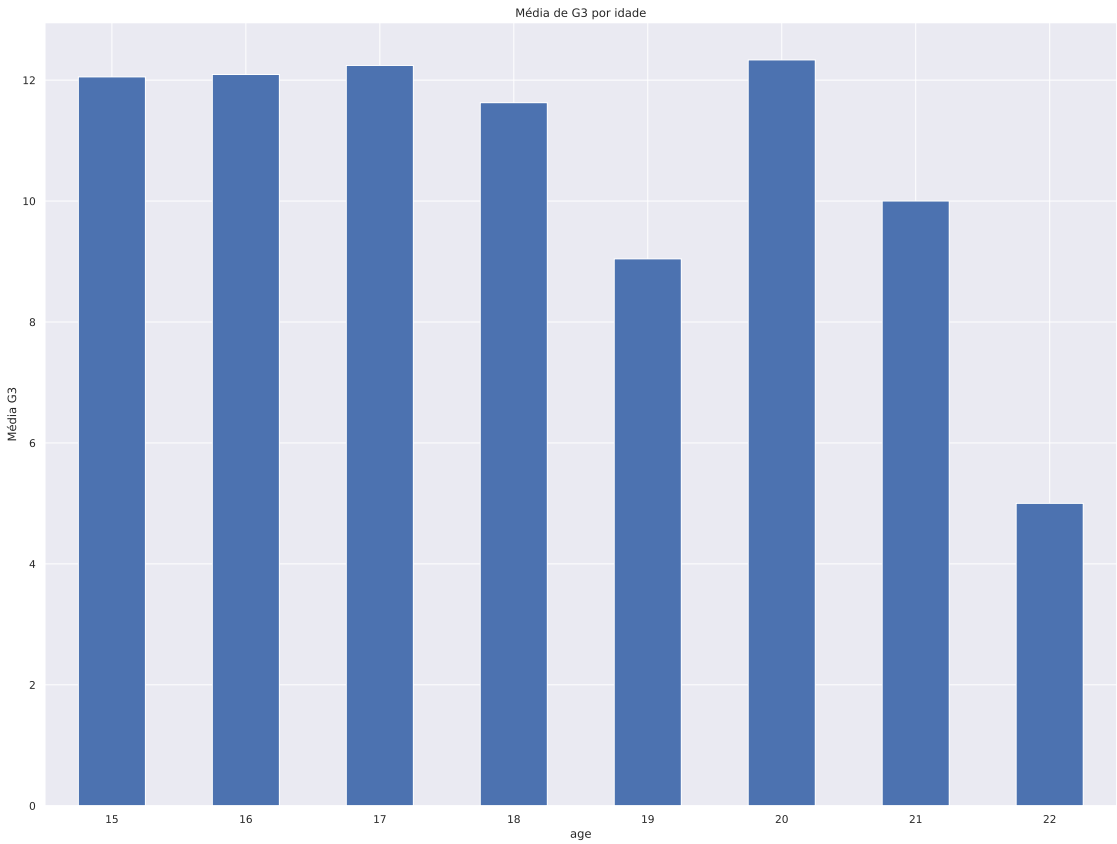


[163]:
```python
por.groupby('age')['G3'].mean().plot(kind='bar')
plt.title('Média de G3 por idade')
plt.ylabel('Média G3')
plt.xticks(rotation=0)
plt.show()
```



[164]:

```
train_por.groupby('age')['G3'].mean().plot(kind='bar')
plt.title('Média de G3 por idade')
plt.ylabel('Média G3')
plt.xticks(rotation=0)
plt.show()

# Apesar das diferenças de distribuição, percebe-se que a relação idade-nota␣
↪segue o mesmo padrão no conjunto total e no conjunto de dados:
# Notas similares de 15 até 18, queda em 19, subida em 20 e queda em 21 e 22.
```



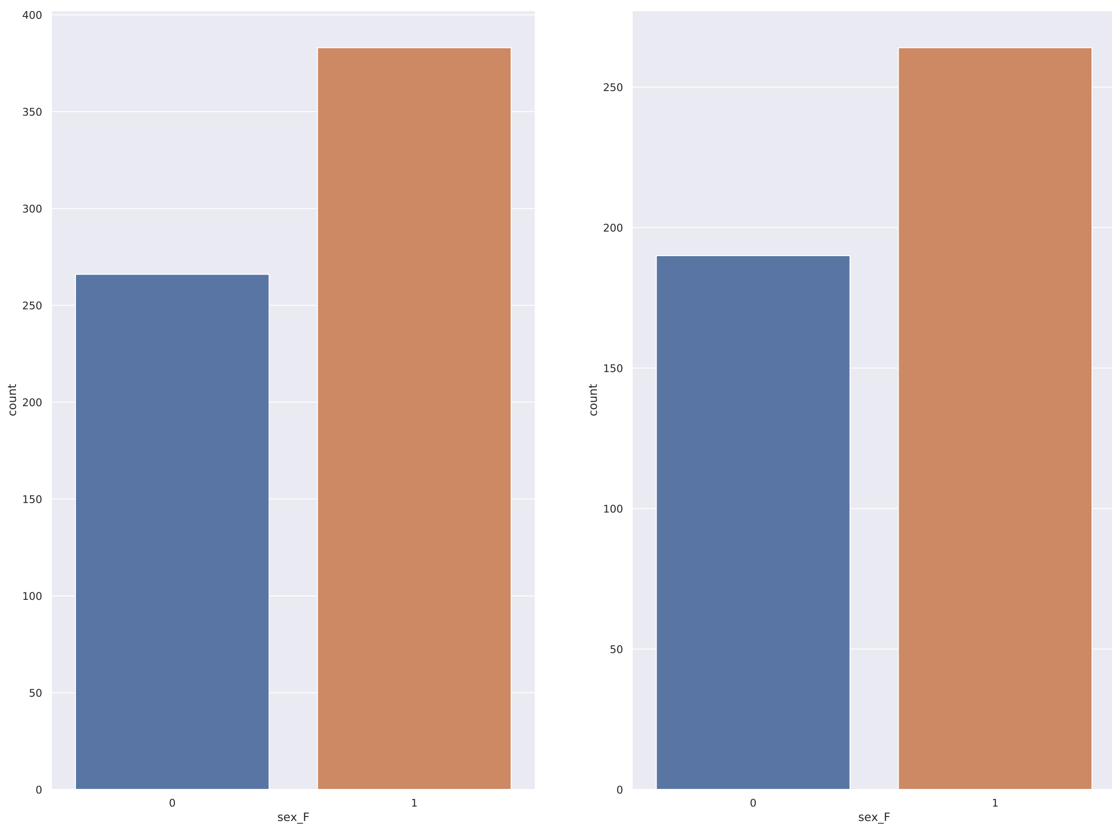
[165]:

```
# Agora vou fazer o mesmo para gênero.

plt.figure(figsize=(15,6))
fig, ax =plt.subplots(1,2)
sns.countplot(por['sex_F'], ax=ax[0])
sns.countplot(train_por['sex_F'], ax=ax[1])
fig.show()
```

```
# Novamente, há mais mulheres do que homens.
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. 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
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. 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
```

```
<Figure size 1080x432 with 0 Axes>
```



[166]:
```
por.groupby('sex_F')[['G3']].mean()
```

[166]:

sex_F	G3
0	11.41
1	12.25

[167]:
```python
train_por.groupby('sex_F')[['G3']].mean()

# Aqui, as coisas se invertem: a média de mulheres é mais alta que a de homens,␣
↪tanto no conjunto total quanto no de treino.
```

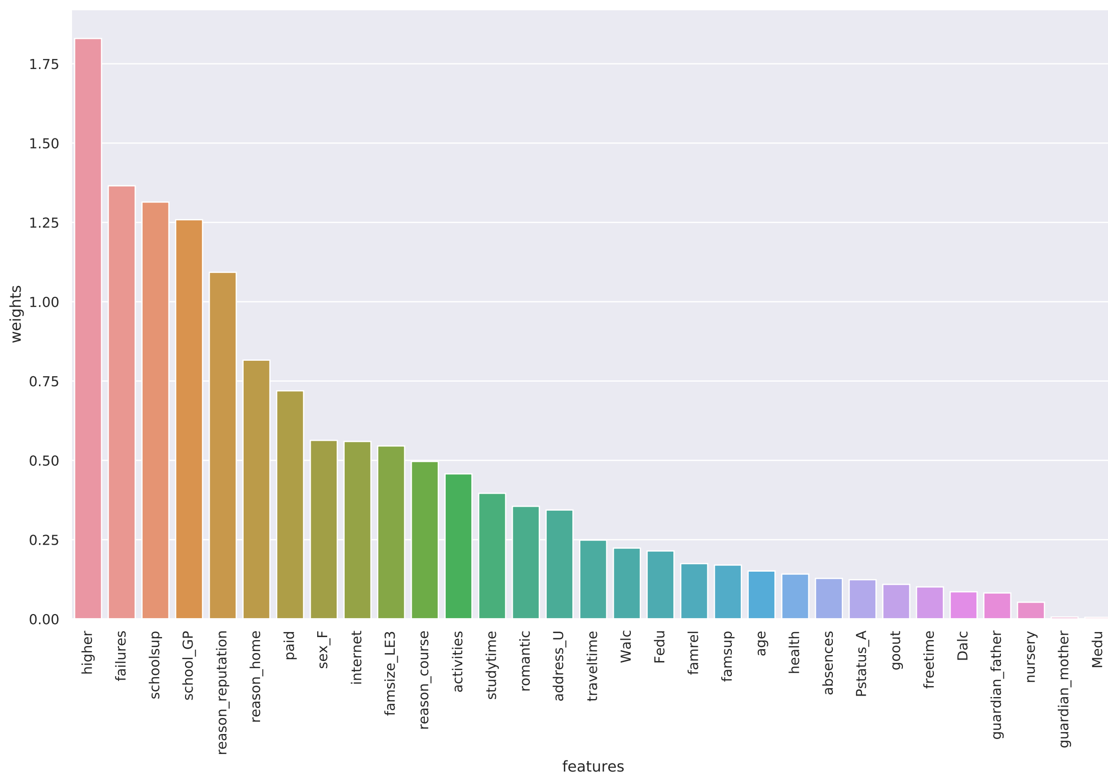
[167]:
```
          G3
sex_F
0      11.35
1      12.20
```

[168]:
```python
# Agora, os coeficientes:
features_t_imp = train_por.copy().drop(['G3'], axis=1)
target_t_imp = train_por.copy()['G3']
```

[169]:
```python
scaler_num = StandardScaler()
features_t_imp[['age', 'absences']] = scaler_num.
↪fit_transform(features_t_imp[['age', 'absences']])
```

[170]:
```python
linear_regressor = LinearRegression()
linear_regressor.fit(features_t_imp, target_t_imp)

feature_importances_lr_coef = pd.concat([pd.Series(features_t_imp.columns,␣
↪name='features'),
                                         pd.Series(linear_regressor.coef_,␣
↪name='weights')],
                                        axis=1)

feature_importances_lr_coef['weights'] =␣
↪abs(feature_importances_lr_coef['weights'])
feature_importances_lr_coef = feature_importances_lr_coef.
↪sort_values(by='weights', ascending=False).reset_index(drop=True)
```

[171]:
```python
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_lr_coef, x='features', y='weights')
plt.xticks(rotation=90)
plt.show()
```



O peso das variáveis tem a seguintes 5 top features:

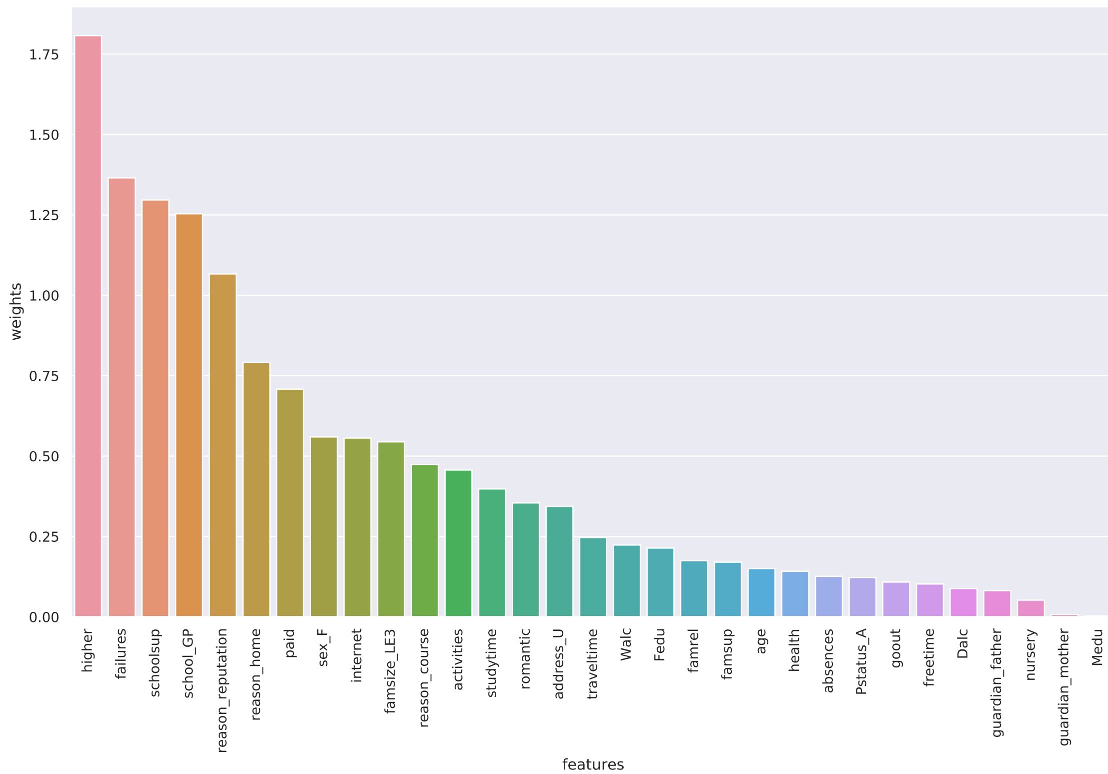
- Querer educação superior
- Failures passados
- Suplemento escolar
- Ser da escola GP
- Ter escolhido a escola pela sua reputação

E se usarmos outro tipo de regressão?

[172]:
```
# Ridge

ridge_reg = Ridge(alpha = 0.5)
ridge_reg.fit(features_t_imp, target_t_imp)

feature_importances_rr_coef = pd.concat([pd.Series(features_t_imp.columns,␣
 ↪name='features'),
                                          pd.Series(ridge_reg.coef_,␣
 ↪name='weights')],
                                         axis=1)

feature_importances_rr_coef['weights'] =␣
 ↪abs(feature_importances_rr_coef['weights'])
```

```
feature_importances_rr_coef = feature_importances_rr_coef.
↪sort_values(by='weights', ascending=False).reset_index(drop=True)
```

[173]:
```
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_rr_coef, x='features', y='weights')
plt.xticks(rotation=90)
plt.show()
```



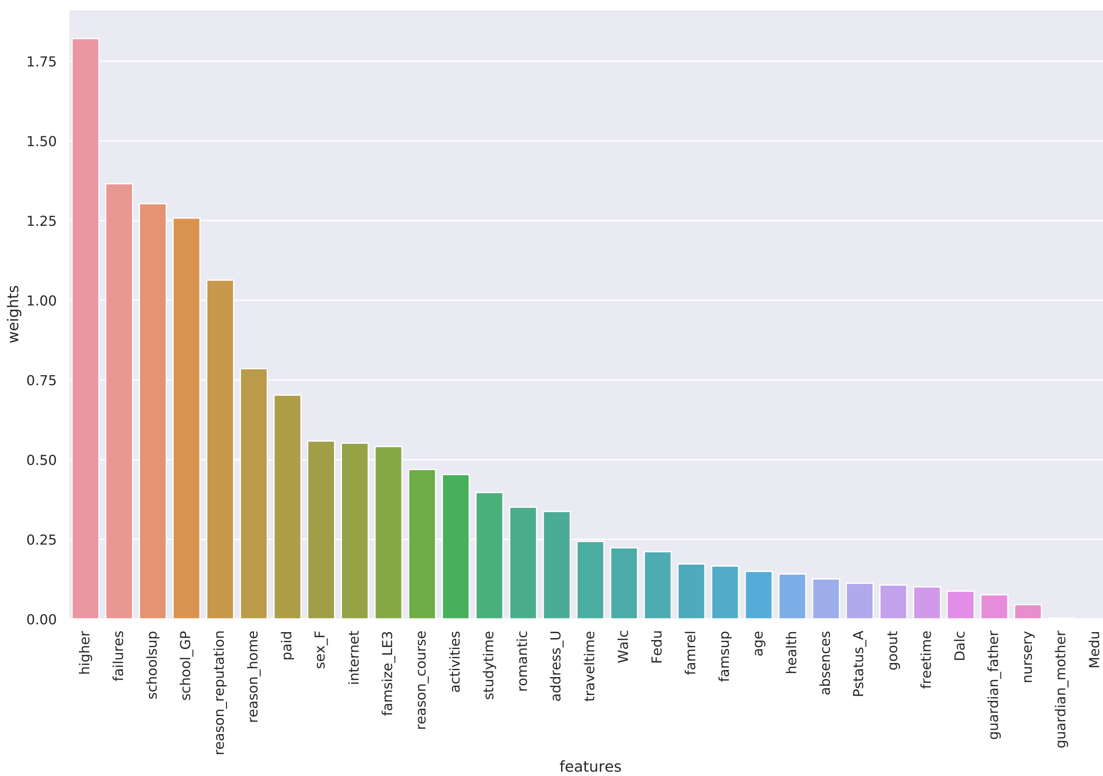
Ridge nos da as seguintes top 5 features:

- querer educação superior
- failures passados
- suplemento escolar
- ser da escola GP
- ter escohido a escola por sua reputação

[174]:
```
# Lasso

model_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(features_t_imp,␣
↪target_t_imp)
```

```
feature_importances_lr_coef = pd.concat([pd.Series(features_t_imp.columns,␣
 ↪name='features'),
                                          pd.Series(model_lasso.coef_,␣
 ↪name='weights')],
                                         axis=1)

feature_importances_lr_coef['weights'] =␣
 ↪abs(feature_importances_lr_coef['weights'])
feature_importances_lr_coef = feature_importances_lr_coef.
 ↪sort_values(by='weights', ascending=False).reset_index(drop=True)
```

[175]:
```
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_lr_coef, x='features', y='weights')
plt.xticks(rotation=90)
plt.show()
```



Lasso nos da as mesmas top 5 features de ridge:

- querer educação superior
- failures passados
- suplemento escolar
- ser da escola GP
- ter escohido a escola por sua reputação

A diferença aqui é que a regressão zera tanto a educação da mãe quanto a criança estar com a mãe.

10.3 Modelos

Agora testarei, de modo simples, os modelos de melhor encaixe

10.4 Matemática

[176]:
```
x_train = train_mat.drop(['G3'], 1)
y_train = train_mat['G3']

x_test = test_mat.drop(['G3'], 1)
y_test = test_mat['G3']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  after removing the cwd from sys.path.
```

[177]:
```
from sklearn.metrics import mean_absolute_error as mae

# definindo a métrica de acurácia
# optei por MAE porque esta medida não sofre com a divisão por 0, enquanto MAPE␣
 ↪fica arbitrariamente alto quando há um 0 no y_test por conta da divisão
```

[178]:
```
models = {
          'Decision Tree': DecisionTreeRegressor(),
          'LinearRegresion': LinearRegression(),
          'Random Forest': RandomForestRegressor(),
          'Ridge': RidgeCV(),
          'LASSO': LassoCV(),
          'ElasticNet': ElasticNetCV(),
          'Gradient Boosting': GradientBoostingRegressor(),
         }

for name in models:
    model = models[name]
    model.fit(x_train, y_train)
    prediction = model.predict(x_test)
    print(f'{name}: {mae(prediction, y_test)}')
```

```
Decision Tree: 3.9327731092436973
LinearRegresion: 3.52642757307281
Random Forest: 2.687310924369748
Ridge: 3.292119552550319
LASSO: 3.053897531553256
ElasticNet: 3.0154722766638273
Gradient Boosting: 2.9708496710454075
```

Aqui vemos que Random Forest e Gradient Boosting foram os modelos que tiveram os menores erros.

Uma possibilidade de melhorarmos isso é pegarmos as features de maior importância dentro do treino (como vista acima com os coeficientes) para testar.

Olhei as 5 mais importantes segundo a Reg Linear.

[179]:
```
x_train_2 = x_train[['higher', 'schoolsup', 'romantic', 'failures',␣
 ↪'famsize_LE3']]
x_test_2 =  x_test[['higher', 'schoolsup', 'romantic', 'failures',␣
 ↪'famsize_LE3']]
```

[180]:
```
models = {
          'Decision Tree': DecisionTreeRegressor(),
          'LinearRegresion': LinearRegression(),
          'Random Forest': RandomForestRegressor(),
          'Ridge': RidgeCV(),
          'LASSO': LassoCV(),
          'ElasticNet': ElasticNetCV(),
          'Gradient Boosting': GradientBoostingRegressor(),
         }

for name in models:
    model = models[name]
    model.fit(x_train_2, y_train)
    prediction = model.predict(x_test_2)
    print(f'{name}: {mae(prediction, y_test)}')
```

```
Decision Tree: 3.1846862156888323
LinearRegresion: 3.133124806387355
Random Forest: 3.0865105126766244
Ridge: 3.1083850071741153
LASSO: 3.1293175475364765
ElasticNet: 3.0428420371910723
Gradient Boosting: 3.153267992760938
```

O MAE de Ridge, LinearRegression e Decision Tree melhorara. Enquanto os outros, por mais que pouco, pioraram.

10.5 Português

[181]:
```
x_train = train_por.drop(['G3'], 1)
y_train = train_por['G3']

x_test = test_por.drop(['G3'], 1)
y_test = test_por['G3']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  after removing the cwd from sys.path.
```

[182]:
```
models = {
          'Decision Tree': DecisionTreeRegressor(),
          'LinearRegresion': LinearRegression(),
          'Random Forest': RandomForestRegressor(),
          'Ridge': RidgeCV(),
          'LASSO': LassoCV(),
          'ElasticNet': ElasticNetCV(),
          'Gradient Boosting': GradientBoostingRegressor(),
         }

for name in models:
    model = models[name]
    model.fit(x_train, y_train)
    prediction = model.predict(x_test)
    print(f'{name}: {mae(prediction, y_test)}')
```

```
Decision Tree: 2.7948717948717947
LinearRegresion: 2.1299395884781367
Random Forest: 2.1666666666666665
Ridge: 2.113495230648637
LASSO: 2.0819836906193028
ElasticNet: 2.0916806271321926
Gradient Boosting: 2.18224403639899
```

[183]:
```
# Agora com a top 5

x_train_2 = x_train[['higher', 'schoolsup', 'failures', 'school_GP',␣
 ↪'reason_reputation']]
x_test_2 =  x_test[['higher', 'schoolsup', 'failures', 'school_GP',␣
 ↪'reason_reputation']]
```

[184]:
```
models = {
          'Decision Tree': DecisionTreeRegressor(),
          'LinearRegresion': LinearRegression(),
          'Random Forest': RandomForestRegressor(),
          'Ridge': RidgeCV(),
          'LASSO': LassoCV(),
          'ElasticNet': ElasticNetCV(),
          'Gradient Boosting': GradientBoostingRegressor(),
         }

for name in models:
    model = models[name]
    model.fit(x_train_2, y_train)
    prediction = model.predict(x_test_2)
    print(f'{name}: {mae(prediction, y_test)}')
```

```
Decision Tree: 2.241118462038058
LinearRegresion: 2.1539973732210664
Random Forest: 2.203393332169783
Ridge: 2.154999524672769
LASSO: 2.1553718893945217
ElasticNet: 2.1575457965581895
Gradient Boosting: 2.2171297201984976
```

Aqui, somente Decision Tree e Gradient Boosting melhoraram.

11 Exercício 3

12 Exercício 2 com Random Forest

12.1 Matemática

[185]:
```
features_t_imp = train_mat.copy().drop(['G3'], axis=1)
target_t_imp = train_mat.copy()['G3']
```

[186]:
```
parameters = {'max_depth' : [8, 10, 12, 20],
              'n_estimators' : [200, 250, 300],
              'max_features' : [5, 25, 50],
              'min_samples_split' : [2, 4, 6]}
grid_search = GridSearchCV(estimator = RandomForestRegressor(random_state=42),
                           param_grid = parameters,
                           scoring = 'neg_mean_squared_error',
                           cv = 5)
grid_search.fit(features_t_imp, target_t_imp)
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
180 fits failed out of a total of 540.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
180 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-
packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py",
line 467, in fit
    for i, t in enumerate(trees)
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 1043,
in __call__
    if self.dispatch_one_batch(iterator):
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 861, in
dispatch_one_batch
    self._dispatch(tasks)
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 779, in
_dispatch
    job = self._backend.apply_async(batch, callback=cb)
  File "/usr/local/lib/python3.7/dist-packages/joblib/_parallel_backends.py",
line 208, in apply_async
    result = ImmediateResult(func)
  File "/usr/local/lib/python3.7/dist-packages/joblib/_parallel_backends.py",
line 572, in __init__
    self.results = batch()
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 263, in
__call__
    for func, args, kwargs in self.items]
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 263, in
<listcomp>
    for func, args, kwargs in self.items]
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/fixes.py", line
216, in __call__
    return self.function(*args, **kwargs)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py",
line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line
1320, in fit
    X_idx_sorted=X_idx_sorted,
```

```
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line
308, in fit
    raise ValueError("max_features must be in (0, n_features]")
ValueError: max_features must be in (0, n_features]

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972:
UserWarning: One or more of the test scores are non-finite: [-17.49965206
-17.54727528 -17.51577951 -17.74598379 -17.78781388
 -17.73377782 -17.77765099 -17.80404615 -17.83093663 -16.75983726
 -16.58344868 -16.58284214 -16.68983142 -16.66168274 -16.66022621
 -16.80245175 -16.73402462 -16.79535297          nan          nan
          nan          nan          nan          nan          nan
          nan          nan -17.31642851 -17.37242665 -17.3036592
 -17.71280404 -17.65775506 -17.62886434 -17.89309946 -17.72049766
 -17.71181605 -16.62707503 -16.43398439 -16.45575404 -16.79602541
 -16.73227789 -16.75632931 -16.7670219  -16.63876659 -16.66337038
          nan          nan          nan          nan          nan
          nan          nan          nan          nan -17.22773076
 -17.34331488 -17.22640778 -17.65207796 -17.64228785 -17.63897328
 -17.82435661 -17.73536965 -17.73803425 -16.45478295 -16.32250138
 -16.39991844 -16.64595152 -16.6292294  -16.64075791 -16.73984961
 -16.56233367 -16.60826971          nan          nan          nan
          nan          nan          nan          nan          nan
          nan -17.31777345 -17.46312981 -17.37317759 -17.80888436
 -17.77158543 -17.7674575  -17.67284613 -17.6145494  -17.66195643
 -16.42828456 -16.31338428 -16.35581049 -16.79188034 -16.73195037
 -16.75285498 -16.6288864  -16.51323825 -16.54918811          nan
          nan          nan          nan          nan          nan
          nan          nan          nan]
  category=UserWarning,
```

[186]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(random_state=42),
             param_grid={'max_depth': [8, 10, 12, 20],
                         'max_features': [5, 25, 50],
                         'min_samples_split': [2, 4, 6],
                         'n_estimators': [200, 250, 300]},
             scoring='neg_mean_squared_error')
```

[187]:
```
grid_search.best_params_
```

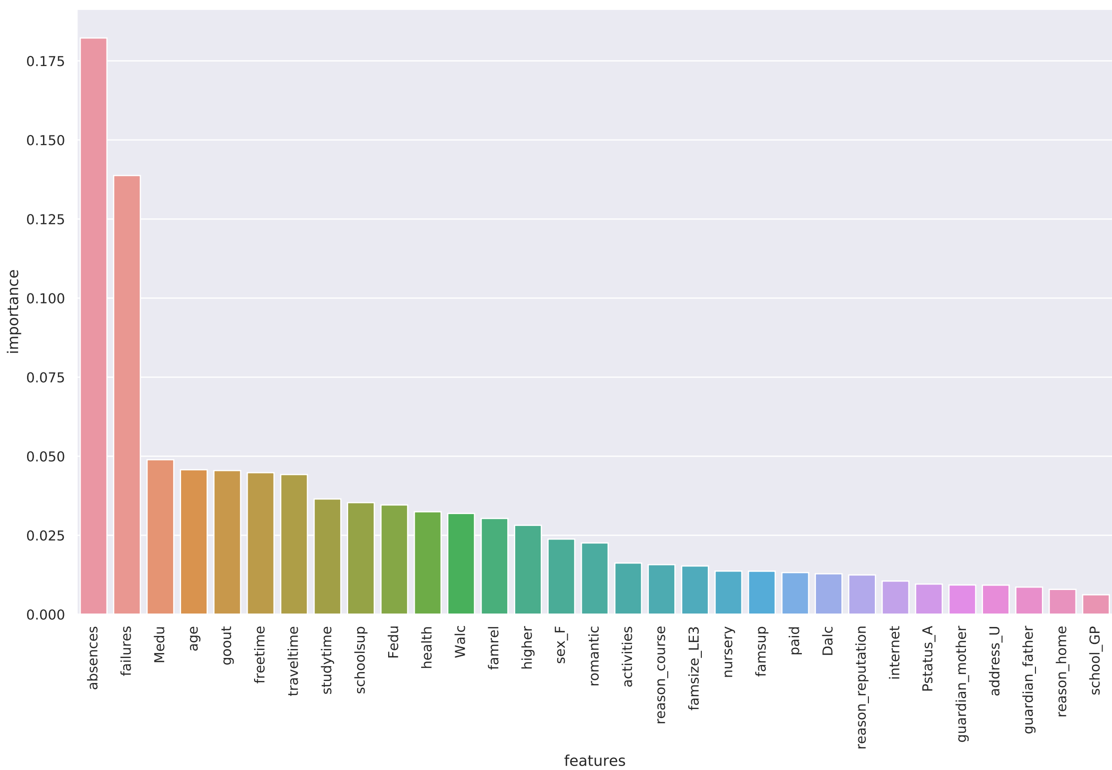
[187]:
```
{'max_depth': 20,
 'max_features': 25,
 'min_samples_split': 2,
 'n_estimators': 250}
```

[188]:
```
regressor_rf = RandomForestRegressor(max_depth=20, n_estimators=250,␣
↪max_features=25, min_samples_split=2, random_state=1)
regressor_rf.fit(features_t_imp, target_t_imp)
```

[188]:
```
RandomForestRegressor(max_depth=20, max_features=25, n_estimators=250,
                      random_state=1)
```

[189]:
```
feature_importances_rf = pd.concat([pd.Series(features_t_imp.columns,␣
↪name='features'),
                                    pd.Series(regressor_rf.
↪feature_importances_, name='importance')],
                                    axis=1).sort_values(by='importance',␣
↪ascending=False).reset_index(drop=True)
```

[190]:
```
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_rf, x='features', y='importance')
plt.xticks(rotation=90)
plt.show()
```



Aqui, as top 5 features são:

- Faltas
- Failures passados

- Educação da mãe
- Idade
- O quanto sai com amigxs

12.2 Será que muda o modelo?

[191]:
```
x_train = train_mat.drop(['G3'], 1)
y_train = train_mat['G3']

x_test = test_mat.drop(['G3'], 1)
y_test = test_mat['G3']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  after removing the cwd from sys.path.
```

[192]:
```
x_train_2 = x_train[['absences', 'failures', 'Medu', 'age', 'goout']]
x_test_2 =  x_test[['absences', 'failures', 'Medu', 'age', 'goout']]
```

[193]:
```
models = {
          'Decision Tree': DecisionTreeRegressor(),
          'LinearRegresion': LinearRegression(),
          'Random Forest': RandomForestRegressor(),
          'Ridge': RidgeCV(),
          'LASSO': LassoCV(),
          'ElasticNet': ElasticNetCV(),
          'Gradient Boosting': GradientBoostingRegressor(),
         }

for name in models:
    model = models[name]
    model.fit(x_train_2, y_train)
    prediction = model.predict(x_test_2)
    print(f'{name}: {mae(prediction, y_test)}')
```

```
Decision Tree: 3.8892156862745098
LinearRegresion: 2.7904687218459623
Random Forest: 2.9413227584740187
Ridge: 2.789705512396568
LASSO: 2.7849384543736404
ElasticNet: 2.783611158131376
Gradient Boosting: 2.9058145565814852
```

O Mae da regressão linear, random forest, ridge, lasso, elastic net e gradient boost melhoram em relação aos modelos com as variáveis mais importantes segundo a regressão linear.

Em relação ao modelo com todas as variáveis, apenas Random Forest piora em acurácia.

12.3 Português

[194]:
```
features_t_imp = train_por.copy().drop(['G3'], axis=1)
target_t_imp = train_por.copy()['G3']
```

[195]:
```
parameters = {'max_depth' : [8, 10, 12, 20],
              'n_estimators' : [200, 250, 300],
              'max_features' : [5, 25, 50],
              'min_samples_split' : [2, 4, 6]}
grid_search = GridSearchCV(estimator = RandomForestRegressor(random_state=42),
                           param_grid = parameters,
                           scoring = 'neg_mean_squared_error',
                           cv = 5)
grid_search.fit(features_t_imp, target_t_imp)
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
180 fits failed out of a total of 540.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
180 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-
packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py",
line 467, in fit
    for i, t in enumerate(trees)
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 1043,
in __call__
    if self.dispatch_one_batch(iterator):
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 861, in
dispatch_one_batch
    self._dispatch(tasks)
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 779, in
_dispatch
    job = self._backend.apply_async(batch, callback=cb)
```

```
  File "/usr/local/lib/python3.7/dist-packages/joblib/_parallel_backends.py",
line 208, in apply_async
    result = ImmediateResult(func)
  File "/usr/local/lib/python3.7/dist-packages/joblib/_parallel_backends.py",
line 572, in __init__
    self.results = batch()
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 263, in
__call__
    for func, args, kwargs in self.items]
  File "/usr/local/lib/python3.7/dist-packages/joblib/parallel.py", line 263, in
<listcomp>
    for func, args, kwargs in self.items]
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/fixes.py", line
216, in __call__
    return self.function(*args, **kwargs)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py",
line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line
1320, in fit
    X_idx_sorted=X_idx_sorted,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line
308, in fit
    raise ValueError("max_features must be in (0, n_features]")
ValueError: max_features must be in (0, n_features]

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972:
UserWarning: One or more of the test scores are non-finite: [-6.72693597
-6.70826288 -6.716762   -6.77809082 -6.75807606 -6.78878846
 -6.81998251 -6.76379198 -6.7561549  -6.91714114 -6.87220849 -6.84747356
 -6.93620286 -6.89762866 -6.87723754 -6.80334736 -6.79306264 -6.81189099
         nan         nan         nan         nan         nan         nan
         nan         nan         nan -6.72579779 -6.661682   -6.68846798
 -6.80556206 -6.74522751 -6.74525846 -6.73401041 -6.69356824 -6.69932943
 -6.90519467 -6.86073947 -6.84855528 -6.8629806  -6.84360928 -6.84460778
 -6.82131403 -6.81112367 -6.80783697         nan         nan         nan
         nan         nan         nan         nan         nan         nan
 -6.82639498 -6.76514522 -6.77586883 -6.74062606 -6.7021693  -6.70836217
 -6.71370782 -6.66856422 -6.66033286 -6.97713574 -6.92778009 -6.90961431
 -6.90502008 -6.8684368  -6.86108645 -6.83300991 -6.80612448 -6.79663332
         nan         nan         nan         nan         nan         nan
         nan         nan         nan -6.70553115 -6.68813586 -6.68813276
 -6.68073223 -6.6647397  -6.68017659 -6.74674119 -6.67978927 -6.68262065
 -6.89809732 -6.84780502 -6.84830911 -6.93597062 -6.8955307  -6.88494021
 -6.81861264 -6.77731198 -6.75884662         nan         nan         nan
         nan         nan         nan         nan         nan         nan]
  category=UserWarning,
```

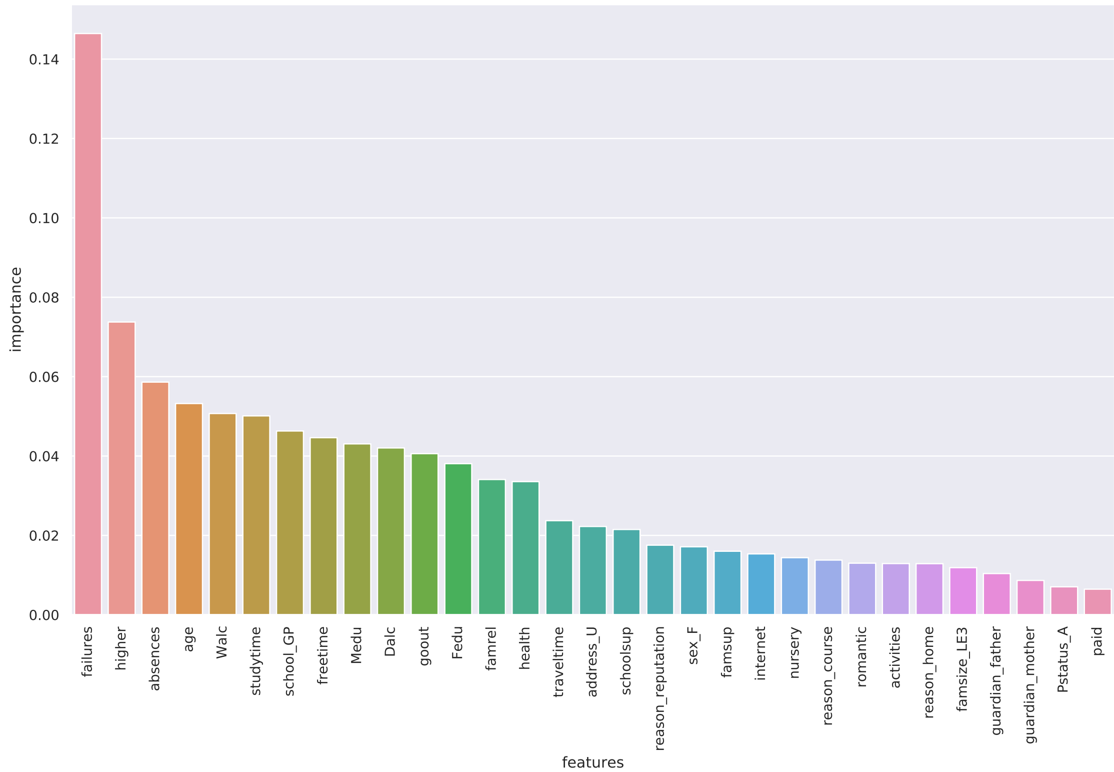
[195]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(random_state=42),
             param_grid={'max_depth': [8, 10, 12, 20],
                         'max_features': [5, 25, 50],
                         'min_samples_split': [2, 4, 6],
                         'n_estimators': [200, 250, 300]},
             scoring='neg_mean_squared_error')
```

[196]:
```
grid_search.best_params_
```

[196]:
```
{'max_depth': 12,
 'max_features': 5,
 'min_samples_split': 6,
 'n_estimators': 300}
```

[197]:
```
regressor_rf = RandomForestRegressor(max_depth=12, n_estimators=300,␣
 ↪max_features=5, min_samples_split=6, random_state=1)
regressor_rf.fit(features_t_imp, target_t_imp)

feature_importances_rf = pd.concat([pd.Series(features_t_imp.columns,␣
 ↪name='features'),
                                    pd.Series(regressor_rf.
 ↪feature_importances_, name='importance')],
                                    axis=1).sort_values(by='importance',␣
 ↪ascending=False).reset_index(drop=True)
```

[198]:
```
plt.figure(figsize=(15,9))
sns.barplot(data=feature_importances_rf, x='features', y='importance')
plt.xticks(rotation=90)
plt.show()
```



Aqui, as top 5 features são

- Failures passados
- Querer educação superior
- Faltas
- Idade
- Consumo de álcool em fins de semana

12.4 Será que muda o modelo?

[199]:
```
x_train = train_por.drop(['G3'], 1)
y_train = train_por['G3']

x_test = test_por.drop(['G3'], 1)
y_test = test_por['G3']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
```

```
argument 'labels' will be keyword-only
  after removing the cwd from sys.path.
```

[200]:
```
x_train_2 = x_train[['absences', 'failures', 'higher', 'age', 'Walc']]
x_test_2 =  x_test[['absences', 'failures', 'higher', 'age', 'Walc']]
```

[201]:
```
models = {
          'Decision Tree': DecisionTreeRegressor(),
          'LinearRegresion': LinearRegression(),
          'Random Forest': RandomForestRegressor(),
          'Ridge': RidgeCV(),
          'LASSO': LassoCV(),
          'ElasticNet': ElasticNetCV(),
          'Gradient Boosting': GradientBoostingRegressor(),
         }

for name in models:
    model = models[name]
    model.fit(x_train_2, y_train)
    prediction = model.predict(x_test_2)
    print(f'{name}: {mae(prediction, y_test)}')
```

```
Decision Tree: 2.805982905982906
LinearRegresion: 2.254832415020636
Random Forest: 2.531450722476686
Ridge: 2.255384276262717
LASSO: 2.255926719662426
ElasticNet: 2.2567412893828505
Gradient Boosting: 2.3823589004453676
```

Em relação aos modelos com todas as variáveis, somente Decision Tree tem sua acurácia melhorada.

Em relação ao modelo com as principais variáveis da regressão linear, nenhuma acurácia é melhorada.

Em conclusão, a acurácia dos modelos não foi profundamente melhorada. Entretanto, é notável que há uma diferença na importância das variáveis dependendo do método utilizado: regressão aponta para um conjunto de características importantes diferente do que é apontando por random forest. Dentre as top 5, compartilham 3 variáveis - failures, absences e age. Entretanto, a classificação de importância das demais variáveis é diferente nos dois métodos.