ANÁLISE DOS MODELOS DE MACHINE LEARNING

Andre Barbosa

Data & Applied Scientist

Microsoft

LinkedIn: *@barbosaandre*
E-Mail: *abarbosa0494@gmail.com*

AGENDA

- **“Análise dos modelos”** (entendendo alguns modelos existentes)
- **“O que é e como explicar uma Caixa Preta”**
- **“Permutation Importance”**
- **“Modelo Caixa de Vidro Vs. Modelo Caixa Preta”** (entendendo os modelos de Caixa Preta e Caixa de Vidro)

Análise dos Modelos

Entendendo alguns modelos existentes

Analisando o código

```
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.metrics import r2_score

X_train, X_test, y_train, y_test = train_test_split(
    dataset[numerical_features], dataset[label], random_state=42, train_size=0.7
)
```

Analisando o código

```
imp_median = SimpleImputer(missing_values=np.nan, strategy="median")
linear_model = Ridge(alpha=0.3)
```

[9] ✓ 0.1s Python

+ Code + Markdown

```
X_train_transformed = imp_median.fit_transform(X_train)
X_test_transformed = imp_median.transform(X_test)

linear_model.fit(X_train_transformed, y_train)
```

[21] ✓ 0.7s Python

```
Ridge(alpha=0.3)
```

```
r2_score(y_test, linear_model.predict(X_test_transformed))
```

[25] ✓ 0.1s Python

```
0.7315371315089919
```

```
    results = cross_validate(
        linear_model,
        X=X_train_transformed,
        y=y_train,
        cv=10,
        scoring=["r2", "neg_root_mean_squared_error"],
    )
    results['test_r2'].mean()
```

4] ✓ 0.1s

```
0.6841001092978575
```

Repetimos o processo via cross-validation

Podemos ver que os resultados foram relativamente próximos :)

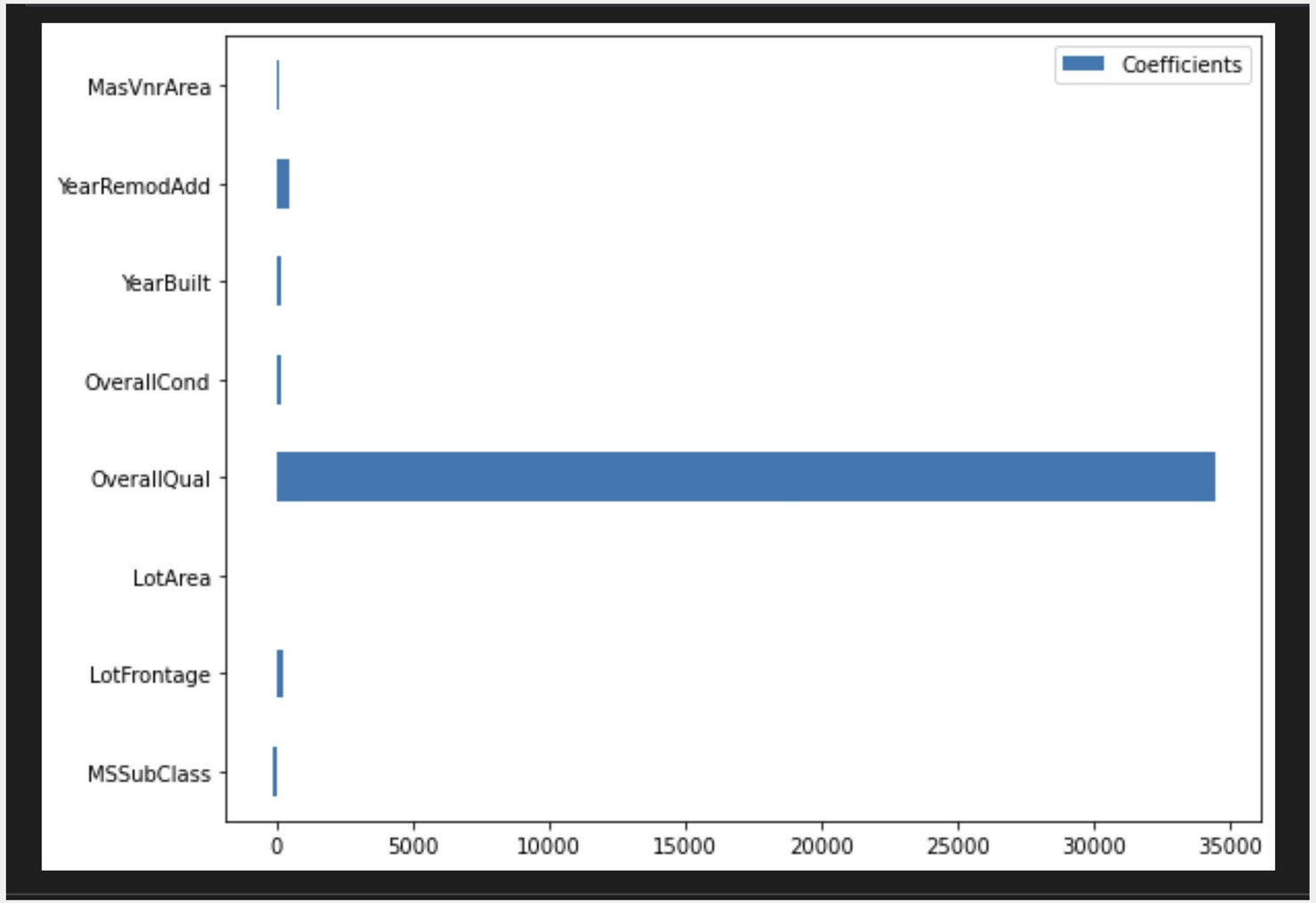
```
trained_model = linear_model
coefs = pd.DataFrame(
    trained_model.coef_.T,
    columns=["Coefficients"], index=numerical_features
)

coefs
```

✓ 0.1s

	Coefficients
MSSubClass	-144.366273
LotFrontage	200.280294
LotArea	1.067949
OverallQual	34459.211810
OverallCond	158.477464
YearBuilt	141.069522
YearRemodAdd	444.328544
MasVnrArea	56.610557

E calculamos os coeficientes!

Pare alguns minutos e reflita sobre o que esses valores querem dizer



O que esses coeficientes querem dizer?

Novamente, pare alguns minutos e reflita!

OverallQual: Rates the overall material and finish of the house

```
10   Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

Relembrando o Dataset

Logo, temos uma variável categórica ordinal! Um aumento no valor da categoria tem relação numérica com o valor esperado

MSSubClass: Identifies the type of dwelling (*habitação/moradia*) involved in the sale.

```
 20  1-STORY 1946 & NEWER ALL STYLES
 30  1-STORY 1945 & OLDER
 40  1-STORY W/FINISHED ATTIC ALL AGES
 45  1-1/2 STORY - UNFINISHED ALL AGES
 50  1-1/2 STORY FINISHED ALL AGES
 60  2-STORY 1946 & NEWER
 70  2-STORY 1945 & OLDER
 75  2-1/2 STORY ALL AGES
 80  SPLIT OR MULTI-LEVEL
 85  SPLIT FOYER
 90  DUPLEX - ALL STYLES AND AGES
120  1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150  1-1/2 STORY PUD - ALL AGES
160  2-STORY PUD - 1946 & NEWER
180  PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190  2 FAMILY CONVERSION - ALL STYLES AND AGES
```

Relembrando o Dataset

Logo, temos uma variável categórica **nominal**! Um aumento no valor da categoria **não** tem relação numérica com o valor de saída. Talvez um processo de Feature Engineering bacana seria fazer o OHE dessas categorias

Analisando o código

```
from sklearn.tree import DecisionTreeRegressor, plot_tree

dt = DecisionTreeRegressor(random_state=42)

dt.fit(X_train_transformed, y_train)
```

Analisando o código

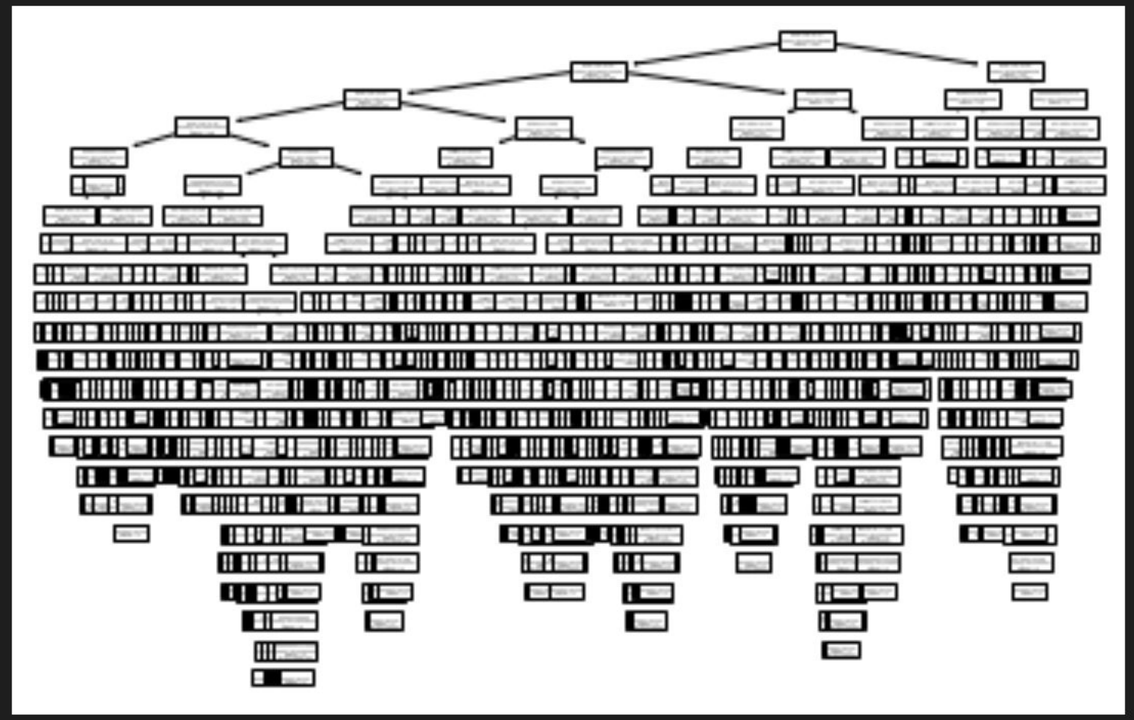
```
[18]  r2_score(y_test, dt.predict(X_test_transformed))
                                                        Python
...   0.6560666533906545
```

```
[21]  results = cross_validate(
          dt,
          X=X_train_transformed,
          y=y_train,
          cv=10,
          scoring=["r2", "neg_root_mean_squared_error"],
      )
      results['test_r2'].mean()
                                                        Python
...   0.5460397023052359
```

Interpretando a árvore

```
[22]  _ = plot_tree(dt, feature_names=numerical_features)
```

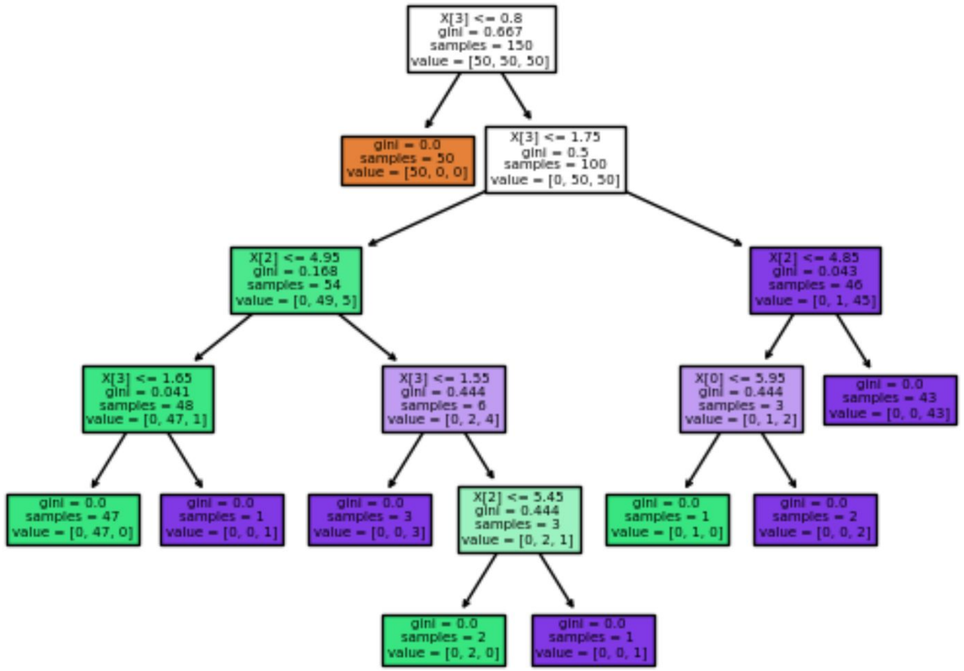
...



WAIT, WHAT?

Mas como interpretar essa imagem?

Árvores de decisão não são tão fáceis de visualizar *em todos os casos*



Interpretabilidade em Decision Trees

Árvores de Decisão são “viáveis” de interpretáveis quando sua altura é baixa. Veja que elas continuam sendo um modelo caixa de vidro!

O que é e como explicar os modelos caixa preta

Analisando o código

```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(random_state=42)


rf.fit(X_train_transformed, y_train)
```

Analisando o código

```
[25]
r2_score(y_test, rf.predict(X_test_transformed))
```

Python

```
0.8153213136306386
```

```
[26]
results = cross_validate(
    rf,
    X=X_train_transformed,
    y=y_train,
    cv=10,
    scoring=["r2", "neg_root_mean_squared_error"],
)
results['test_r2'].mean()
```

Python

```
0.7252432198063722
```

Melhor que a regressão!

```
r2_score(y_test, rf.predict(X_test_transformed))
```

```
0.8153213136306386
```

```
results = cross_validate(
    rf,
    X=X_train_transformed,
    y=y_train,
    cv=10,
    scoring=["r2", "neg_root_mean_squared_error"],
)
results['test_r2'].mean()
```

```
0.7252432198063722
```

```
r2_score(y_test, linear_model.predict(X_test_transformed))
```

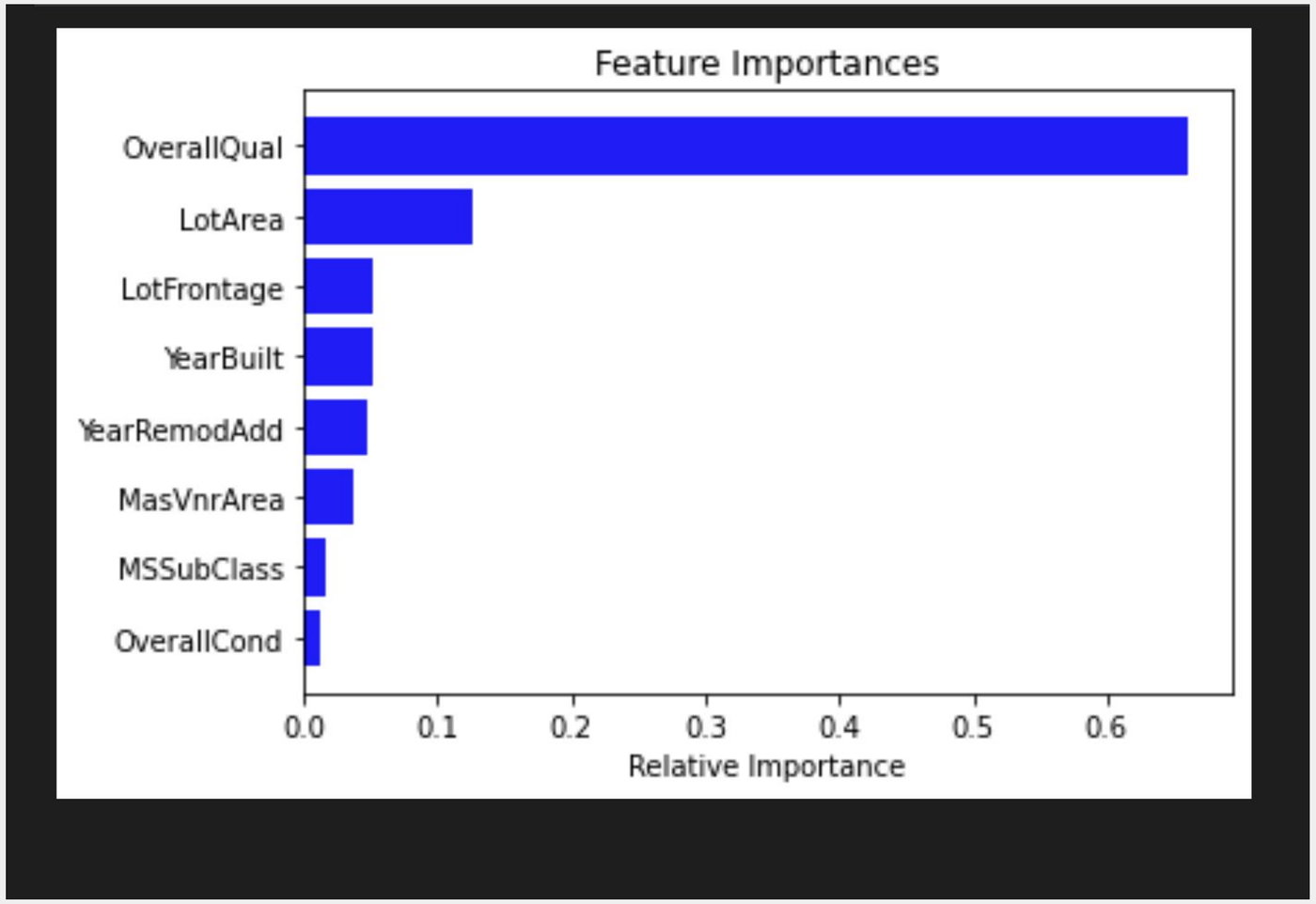
✓ 0.1s

```
0.7315371315089919
```

```
results = cross_validate(
    linear_model,
    X=X_train_transformed,
    y=y_train,
    cv=10,
    scoring=["r2", "neg_root_mean_squared_error"],
)
results['test_r2'].mean()
```

✓ 0.1s

```
0.6841001092978575
```



E podemos gerar noção de importância!

Mas o que isso quer dizer no caso de um algoritmo de Bagging? E no caso de um Boosting?

Permutation Importance

Permutation Importance

"A ideia é a seguinte: a importância do recurso pode ser medida olhando para o quanto os scores (precisão, F1, Rˆ2, etc. - qualquer score em que estejamos interessadas) diminuem quando um recurso está indisponível."

"Para fazer isso, é possível remover o recurso do dataset, retreinar o estimator e checar os scores. Mas isso requer um retreinamento do estimator para cada recurso, o que pode ser computacionalmente intenso. Isso também mostra o que pode ser importante dentro de um dataset, e não o que é importante dentro de um modelo treinado concreto."

LEIA MAIS AQUI

Permutation Importance

"Para evitar retreinamento do *estimator,* podemos remover uma variável somente da parte de teste do dataset, e computar o score sem usar essa variável. Não funciona como é, porque *estimators* esperam que a variável esteja presente. Então, ao invés de remover uma variável, podemos substituí-lo por *ruído aleatório* - a coluna de valores continua ali, mas não contém mais informações úteis. Esse método funciona se o *ruído* é retirado da mesma distribuição dos valores do recurso original (de outro modo, o estimator não vai funcionar). A maneira mais simples de conseguir esse ruído é combinar os dados em um recurso, ou seja, use outros exemplos de valores de recurso - é assim que permutation importance é computada"

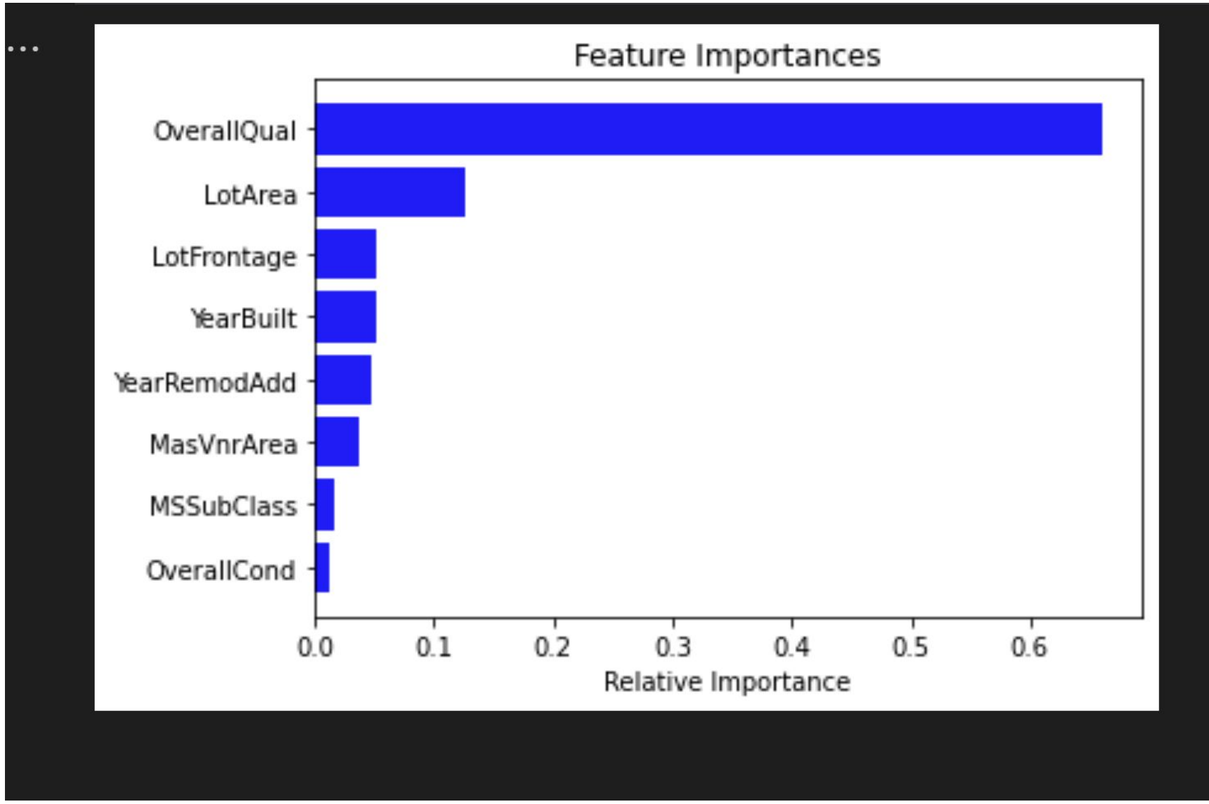
LEIA MAIS AQUI

Os valores são diferentes!

```
import eli5
from eli5.sklearn import PermutationImportance

transform_data = X_test_transformed

perm = PermutationImportance(rf).fit(
    transform_data, y_test
)
eli5.show_weights(perm, feature_names=numerical_features)
```

3]:

Weight	Feature
1.1178 ± 0.1555	OverallQual
0.0922 ± 0.0277	LotArea
0.0249 ± 0.0080	MasVnrArea
0.0182 ± 0.0084	YearBuilt
0.0140 ± 0.0038	LotFrontage
0.0124 ± 0.0072	YearRemodAdd
0.0099 ± 0.0017	MSSubClass
0.0058 ± 0.0049	OverallCond



Modelo Caixa de Vidro VS Modelo Caixa Preta

Modelos Interpretáveis

Modelos interpretáveis são aqueles que **naturalmente** têm alguma interpretabilidade.

Exemplos:
- o peso das variáveis dado através dos coeficientes de uma regressão linear
- O “fluxo” de uma árvore de decisão

Modelos Explicáveis

- São modelos em que um **humano não consegue** entender diretamente.

- Podemos pensar em modelos de ML, em geral, como funções. Essas funções são complexas demais e, para **explicá-las,** nos **aproximamos** por uma outra função, que conseguimos entender!

- Vamos entender mais sobre isso na aula de hoje.

Caixa de Vidro ou Caixa Preta?

Como é possível ver pelas imagens, as classes de modelos de Machine Learning dividem-se em duas:

- modelos de **caixa de vidro**, em que as interpretações/predições são claras e direta;
- modelos **caixa preta**, em que os resultados não são facilmente interpretáveis por um humano.

Importando as bibliotecas

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import warnings



pd.options.display.max_rows = 60
pd.options.display.max_columns = 100
warnings.simplefilter("ignore")
```

Montando o Dataset

```
dataset = pd.read_csv("dataset/house_dataset.csv", index_col=0)
```

Descrição do Dataset:

```
MSSubClass: Identifies the type of dwelling involved in the sale.

        20    1-STORY 1946 & NEWER ALL STYLES
        30    1-STORY 1945 & OLDER
        40    1-STORY W/FINISHED ATTIC ALL AGES
        45    1-1/2 STORY - UNFINISHED ALL AGES
        50    1-1/2 STORY FINISHED ALL AGES
        60    2-STORY 1946 & NEWER
        70    2-STORY 1945 & OLDER
        75    2-1/2 STORY ALL AGES
        80    SPLIT OR MULTI-LEVEL
        85    SPLIT FOYER
        90    DUPLEX - ALL STYLES AND AGES
       120    1-STORY PUD (Planned Unit Development) - 1946 & NEWER
       150    1-1/2 STORY PUD - ALL AGES
       160    2-STORY PUD - 1946 & NEWER
       180    PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
       190    2 FAMILY CONVERSION - ALL STYLES AND AGES
```

MSZoning: Identifies the general zoning classification of the sale.

A	Agriculture
C	Commercial
FV	Floating Village Residential
I	Industrial
RH	Residential High Density
RL	Residential Low Density
RP	Residential Low Density Park
RM	Residential Medium Density

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl	Gravel
Pave	Paved

Montando o Dataset

```
dataset = pd.read_csv("dataset/house_dataset.csv", index_col=0)
```

```
dataset.head()
```

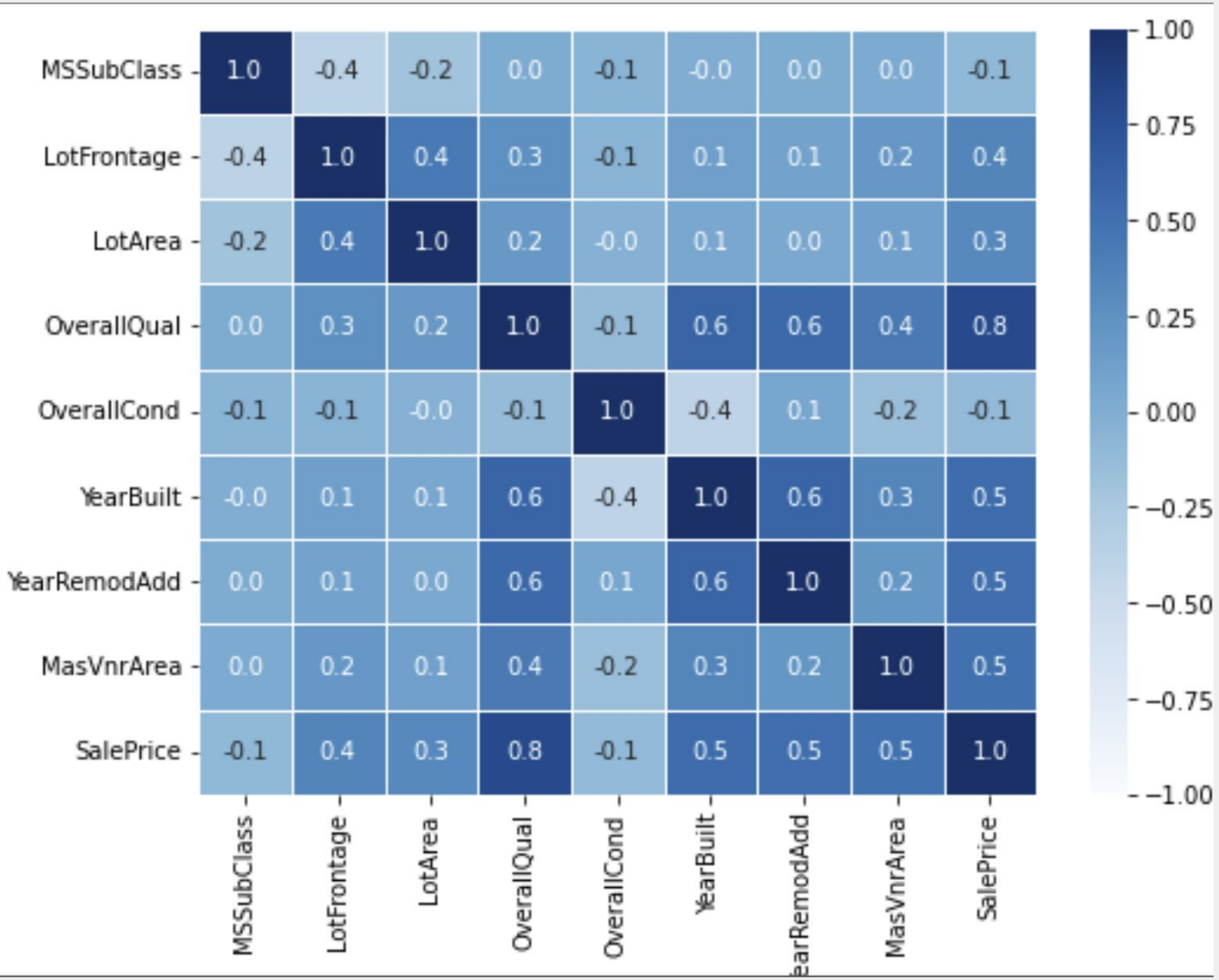
Python

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSl
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	

```python
numerical_features = [
    "MSSubClass",
    "LotFrontage",
    "LotArea",
    "OverallQual",
    "OverallCond",
    "YearBuilt",
    "YearRemodAdd",
    "MasVnrArea",
]
label = "SalePrice"
dataset[numerical_features].isna().sum()
```

```
MSSubClass       0
LotFrontage    259
LotArea          0
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
MasVnrArea       8
dtype: int64
```

Por questões de simplificação, iremos trabalhar _apenas_ com as variáveis numéricas :)



Calculando algumas estatísticas simples

```python
df_corr = dataset[numerical_features + [label]].dropna().corr()
plt.figure(figsize=(8, 6))
sns.heatmap(
    df_corr,
    annot=True,
    fmt=".1f",
    linewidths=0.5,
    cmap="Blues",
    center=0,
    vmax=1.0,
    vmin=-1.0,
)
```