

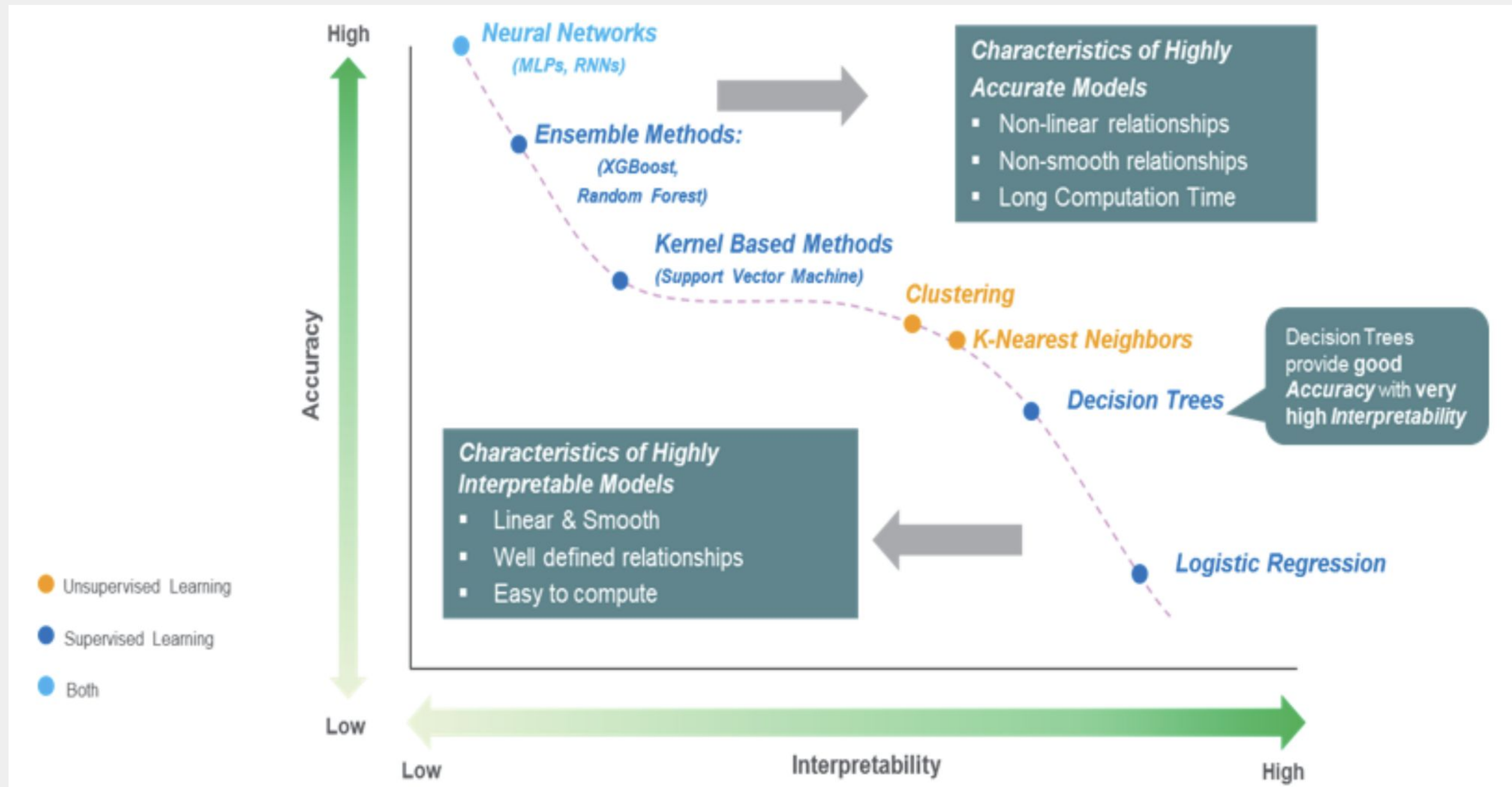
O que são os metamodelos?

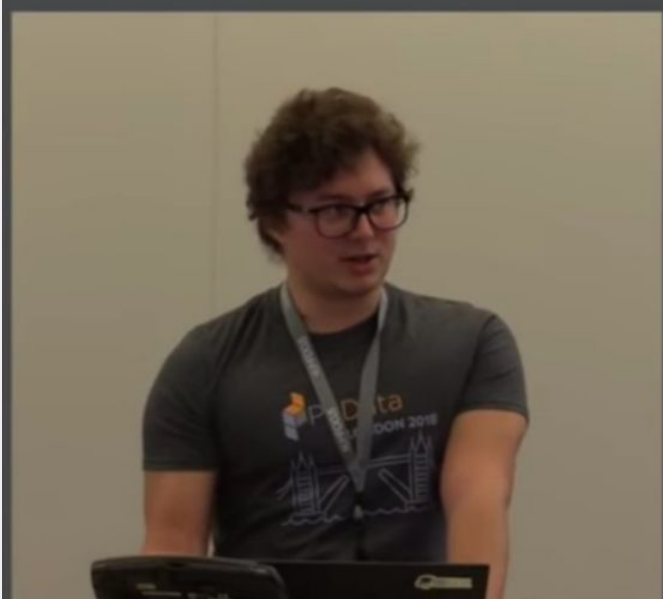
Entendendo os metamodelos e o conceito de Lime e SHAP

A vertical bar with a gradient from bright green at the top to light blue at the bottom.

O conceito de Metamodelo

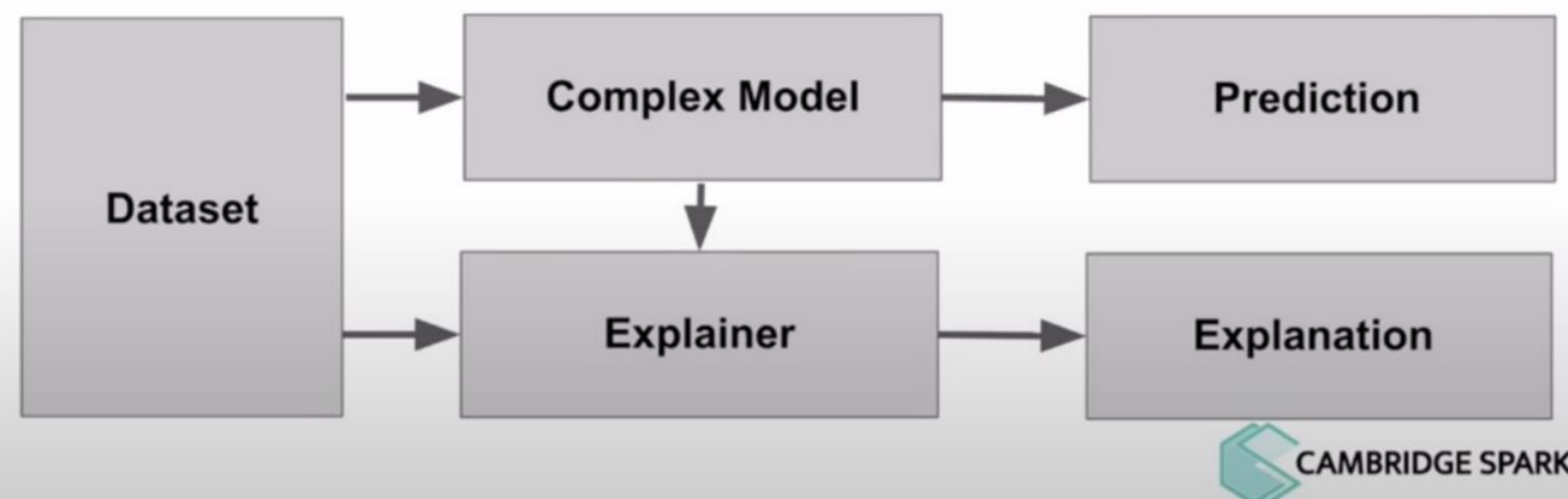
Por muito tempo isso já foi considerado verdade!





Shapley Additive Explanations (SHAP)

- An explanation model is a simpler model that is a good approximation of our complex model.



Metamodelos

E se ao invés de treinarmos os modelos e tentarmos entender eles, tentássemos usar um **outro** modelo para explicar o primeiro ?

Se conseguimos modelos lineares são explicáveis/interpretáveis, o que poderíamos esperar desse novo modelo?

No caso, esse outro modelo seria **linear** e, logo, poderíamos interpretar mais ou menos como estamos acostumados :)

Logo, um primeiro pensamento intuitivo seria aproximar as previsões por uma **regressão linear**. Faz sentido, certo?

T



O conceito LIME

LIME

"Uma explicação é criada aproximando o modelo subjacente local de um modelo interpretável. Modelos interpretáveis são, por exemplo, modelos lineares com forte regularização, árvores de decisão, etc. Os modelos interpretáveis são treinados em pequenas perturbações da instância original e devem somente prover uma boa aproximação local. O dataset é criado, pela adição de ruído a recursos contínuos, removendo palavras ou escondendo partes da imagem. Só de aproximar o black box localmente (na vizinhança das amostras de dados) a tarefa é significativamente simplificada."

LIME

"An explanation is created by approximating the underlying model locally by an interpretable one. Interpretable models are e.g. linear models with strong regularisation, decision trees, etc. The interpretable models are trained on small perturbations of the original instance and should only provide a good local approximation. The 'dataset' is created by e.g. adding noise to continuous features, removing words or hiding parts of the image. By only approximating the black-box locally (in the neighborhood of the data sample) the task is significantly simplified."

LEIA MAIS [AQUI](#)

Analizando o código

```
import lime
import lime.lime_tabular

explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    training_labels=y_train.values,
    feature_names=numerical_features,
    verbose=True,
    mode="regression",
    discretize_continuous=False
)
```

Analizando o LIME

```
In [32]: ▶ exp = explainer.explain_instance(  
        X_test.iloc[0].values, rf.predict)
```

```
Intercept 185684.4727948665  
Prediction_local [181567.32396607]  
Right: 164300.0
```

```
In [33]: ▶ exp.show_in_notebook(show_table=True)
```

Predicted value

69407.00 503505.50
(min) 164300.00 (max)

negative

positive

OverallQual 54180.0
LotArea 25726.10
YearBuilt 5059.69
YearRemodAdd 4568.29
LotFrontage 2565.34
OverallCond 2145.54
MasVnrArea 1902.03
MSSubClass 2.14

Feature Value

OverallQual	6.00
LotArea	8414.00
YearBuilt	1963.00
YearRemodAdd	2003.00
LotFrontage	70.00
OverallCond	8.00
MasVnrArea	0.00
MSSubClass	20.00

Drawbacks do LIME

Apesar de perfeito, existem alguns poréns/limitações, que são discutidas [aqui](#), [aqui](#) e [aqui](#)

Dessas, podemos destacar as seguintes:

1. O LIME só permite explicações locais, não dando uma noção global das features
2. Ele é computacionalmente caro. Considere o mesmo caso levantado com o Permutation Importance
3. A "noção de vizinhança" para gerar as amostras é algo bem custoso e complicado de definir. A implementação do LIME faz amostras de uma distribuição Gaussiana, que assume que as variáveis são sempre independentes e, bem, sabemos que na prática isso nem sempre é verdade

Então beleza, aproximar por uma regressão linear parece problemática, uma vez que dá uma visão só local e tem as suas devidas limitações. O que podemos, então, fazer?



O conceito SHAP



Shapley Values Defined in Game Theory

How to Divide Money Between Game Players in a Fair Way?

Fairness properties:

- Additivity (Amounts sum up to the final game result)
- Consistency (More effect – not less money)

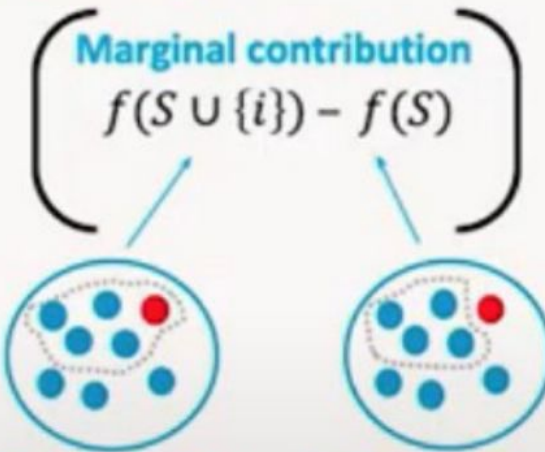
Theorem: The only fair way to distribute the money is:

Shapley Value for player i in game f $=$ Average over all players' subsets $S \subseteq M/\{i\}$

f – game
 M – all players
 S – subset of players
 i – specific player

Marginal contribution
 $f(S \cup \{i\}) - f(S)$

Money difference with and without this player






De Teoria dos Jogos para ML

Como vocês puderam ver, o trabalho em cima do SHAP é bastante simples. Além disso, ele é bastante interessante porque ele **garante** duas propriedades super importantes para a definição de uma distribuição *justa*.

Recursos sobre SHAP

Inclusive, uma dessas propriedades é **violada** no caso do *get_feature_importance* dos modelos de Ensemble famosos. Para quem tiver curiosidade...

Apesar de computar SHAPley Values ser extremamente caro (drawback), existem implementações **específicas** para os casos de Árvore, **GLMs** e **alguns outros casos** graças a biblioteca **SHAP** ! Isso foi o que fez a biblioteca ser utilizada, em produção, por várias empresas. tais como: **Nubank; QuintoAndar; Google e Microsoft**



A relação entre explicabilidade e causalidade

Analizando o código

```
import shap

# load JS visualization code to notebook
shap.initjs()

explainer = shap.TreeExplainer(rf)
shap_values = explainer(X_test)

shap_values = shap.Explanation(shap_values.values,
                             base_values=shap_values.base_values.reshape(-1),
                             data=shap_values.data,
                             feature_names=shap_values.feature_names)
```

Analizando uma Instância

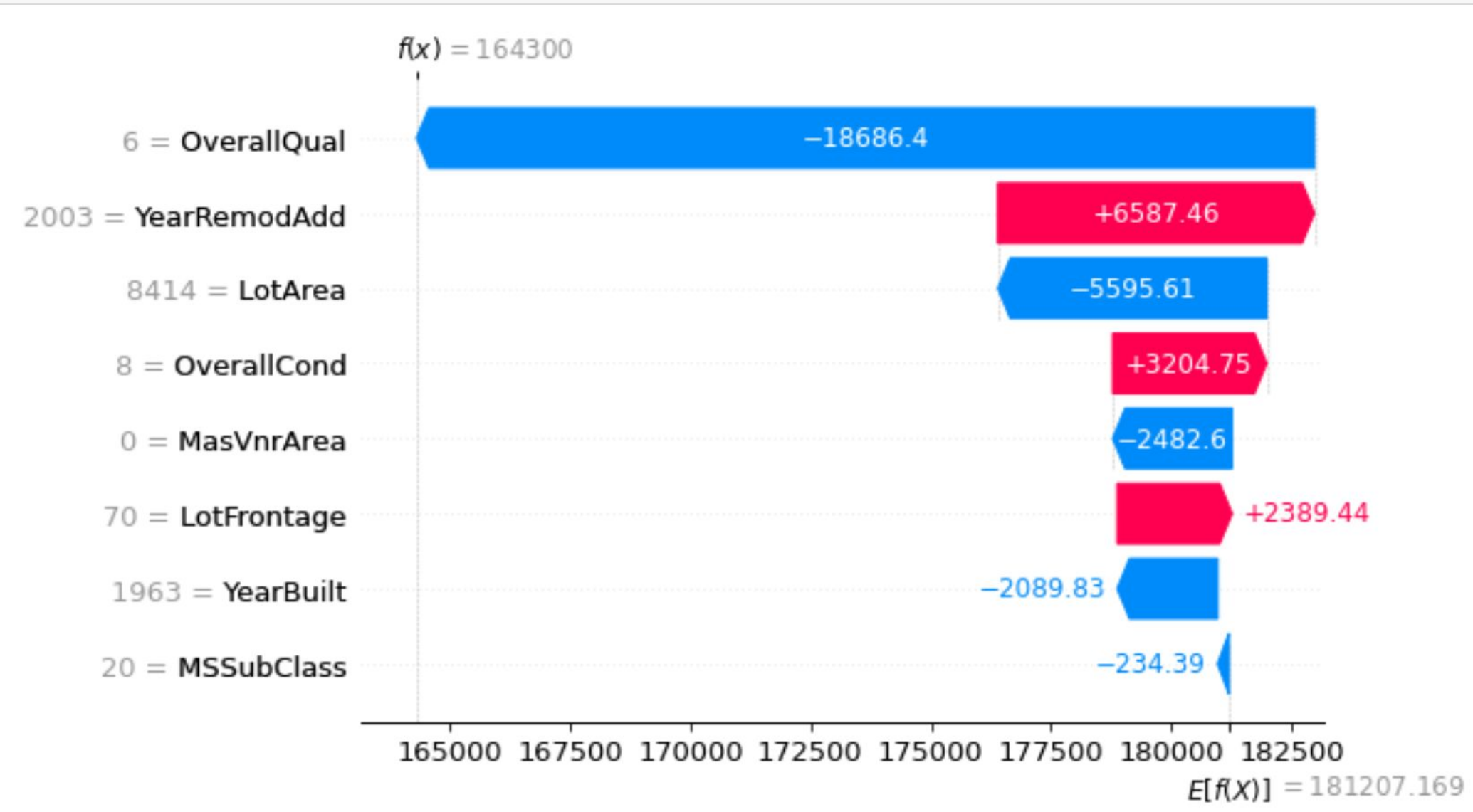
```
shap.plots.force(shap_values[0])
```



T

Analizando uma única instância

```
shap.plots.waterfall(shap_values[0])
```



```
In [39]: rf.predict(X_test_transformed[0, :].reshape(1, -1))
```

```
Out[39]: array([164300.])
```

```
In [40]: shap_values.base_values[0] + shap_values.values[0].sum()
```

```
Out[40]: 164299.999999999983
```

```
In [41]: X_test[['MasVnrArea', 'OverallQual', 'LotArea']].head(1)
```

```
Out[41]:
```

	MasVnrArea	OverallQual	LotArea
Id			
893	0.0	6	8414

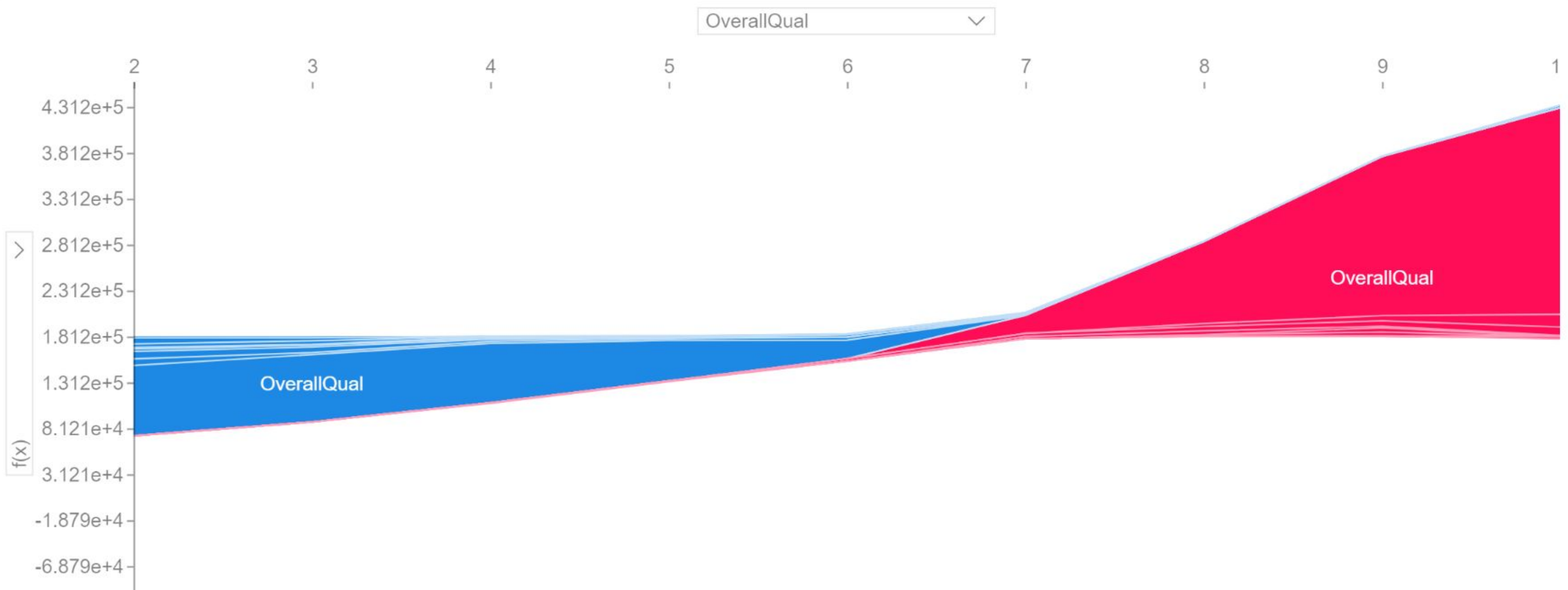
```
In [42]: 1.802e+5, 1.169e+4
```

```
Out[42]: (180200.0, 11690.0)
```

Relações entre variáveis

```
In [43]: ▶ # visualize the training set predictions  
shap.force_plot(explainer.expected_value, shap_values.values, X_test_transformed, feature_names=numerical_features)
```

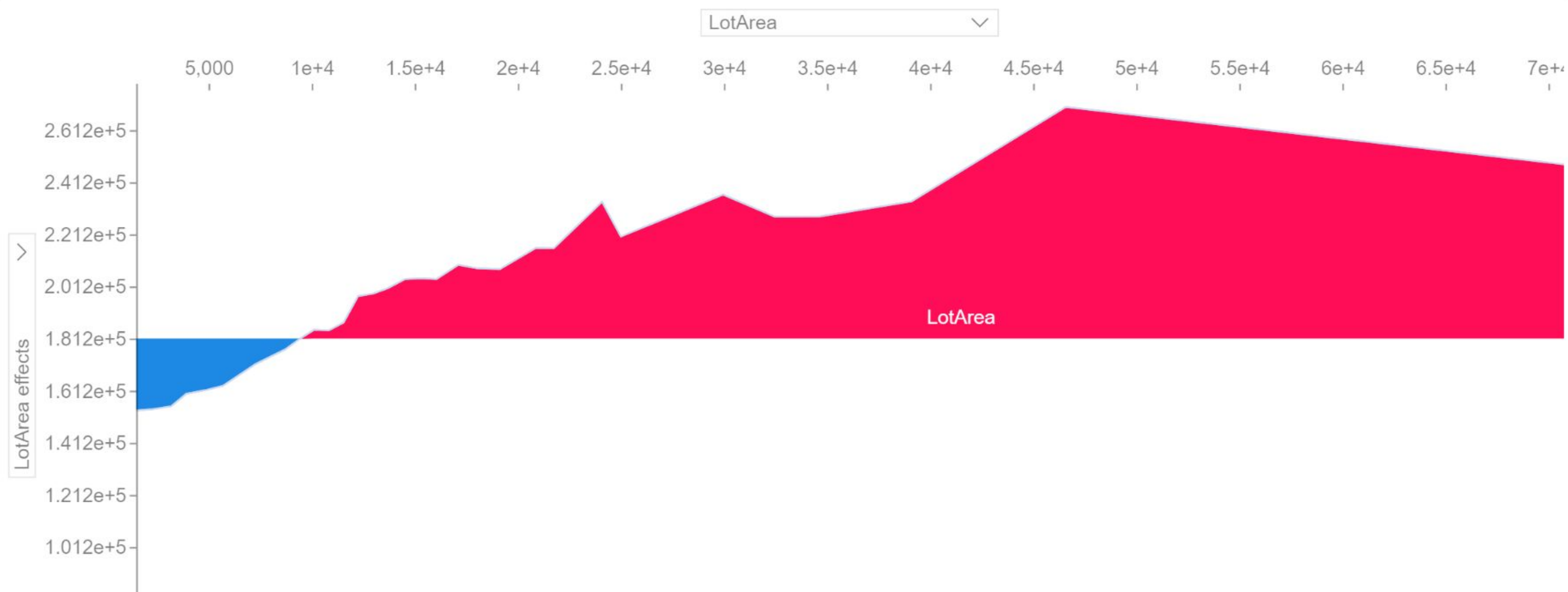
Out[43]:



Relações entre variáveis

```
In [43]: # visualize the training set predictions  
shap.force_plot(explainer.expected_value, shap_values.values, X_test_transformed, feature_names=numerical_features)
```

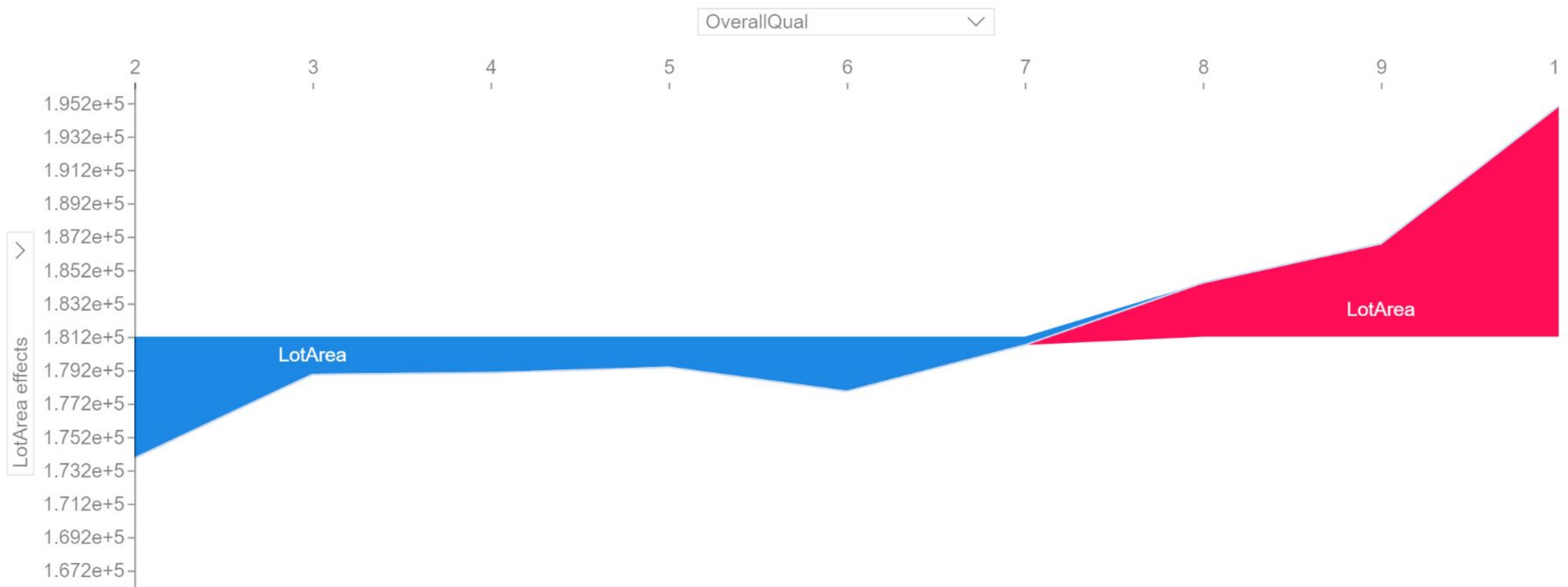
Out[43]:



Relações entre variáveis

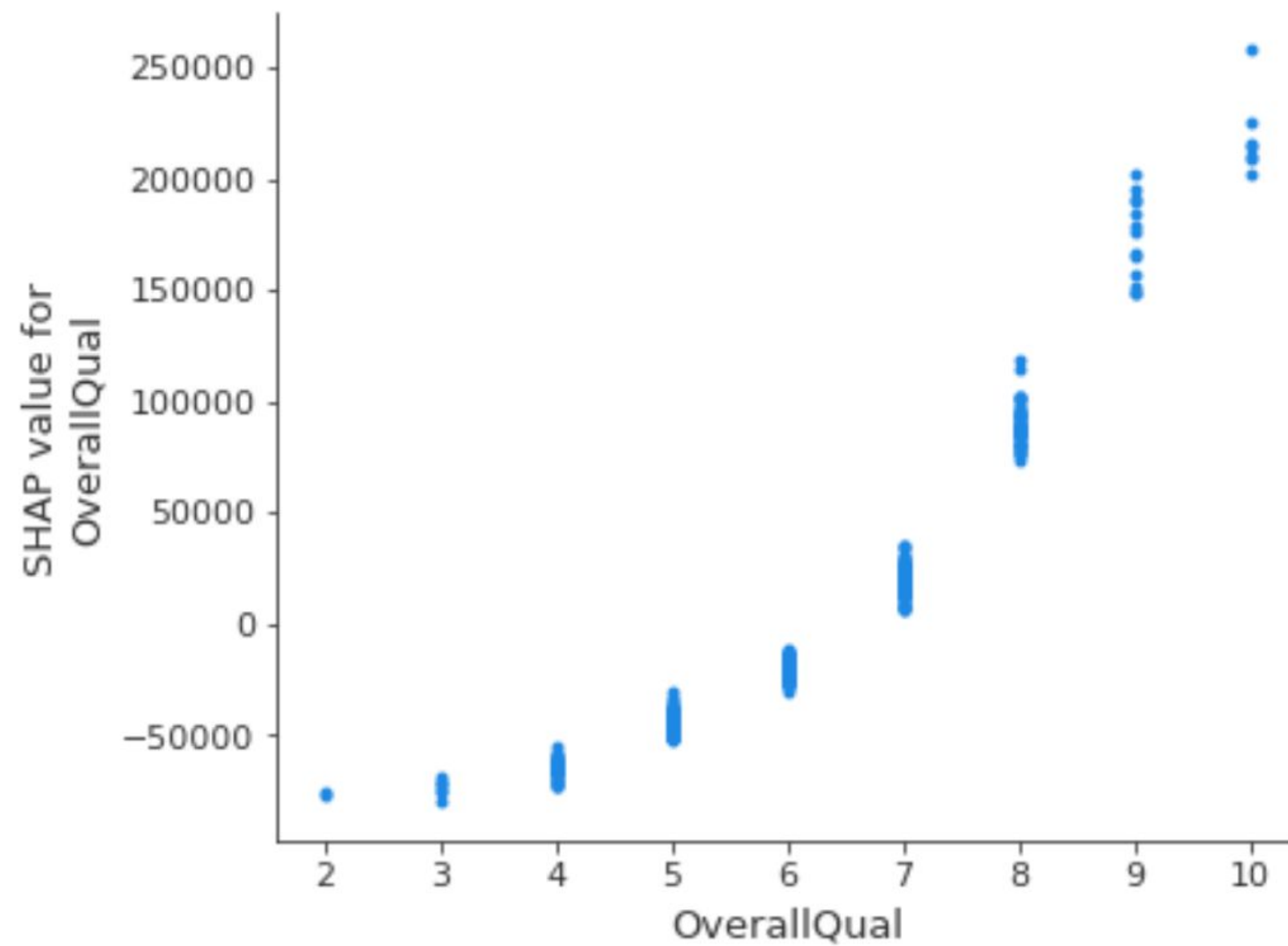
```
In [43]: # visualize the training set predictions  
shap.force_plot(explainer.expected_value, shap_values.values, X_test_transformed, feature_names=numerical_features)
```

Out[43]:



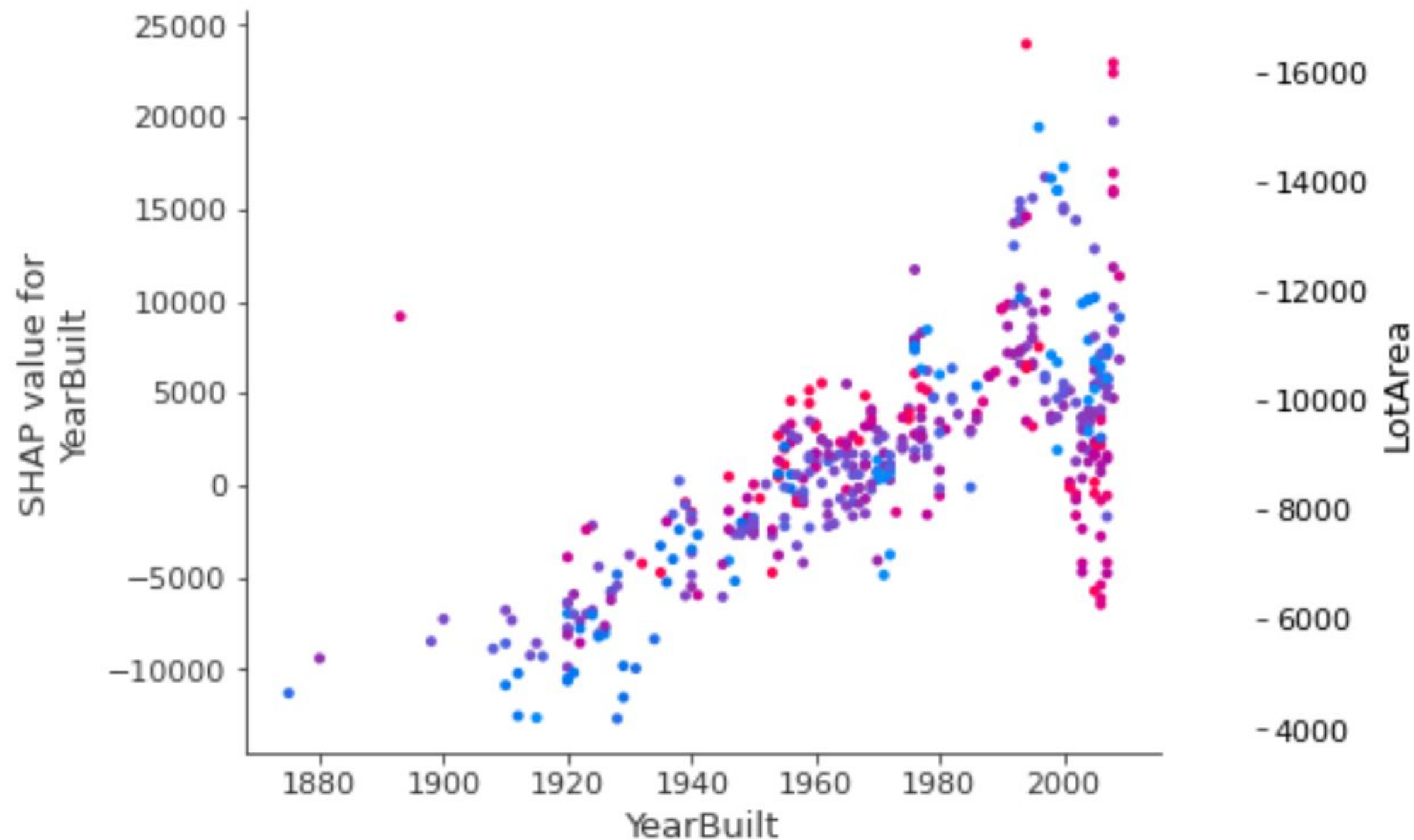
Dependence Plot

```
In [44]: ▶ shap.dependence_plot("OverallQual", shap_values.values, X_test, interaction_index=None)
```



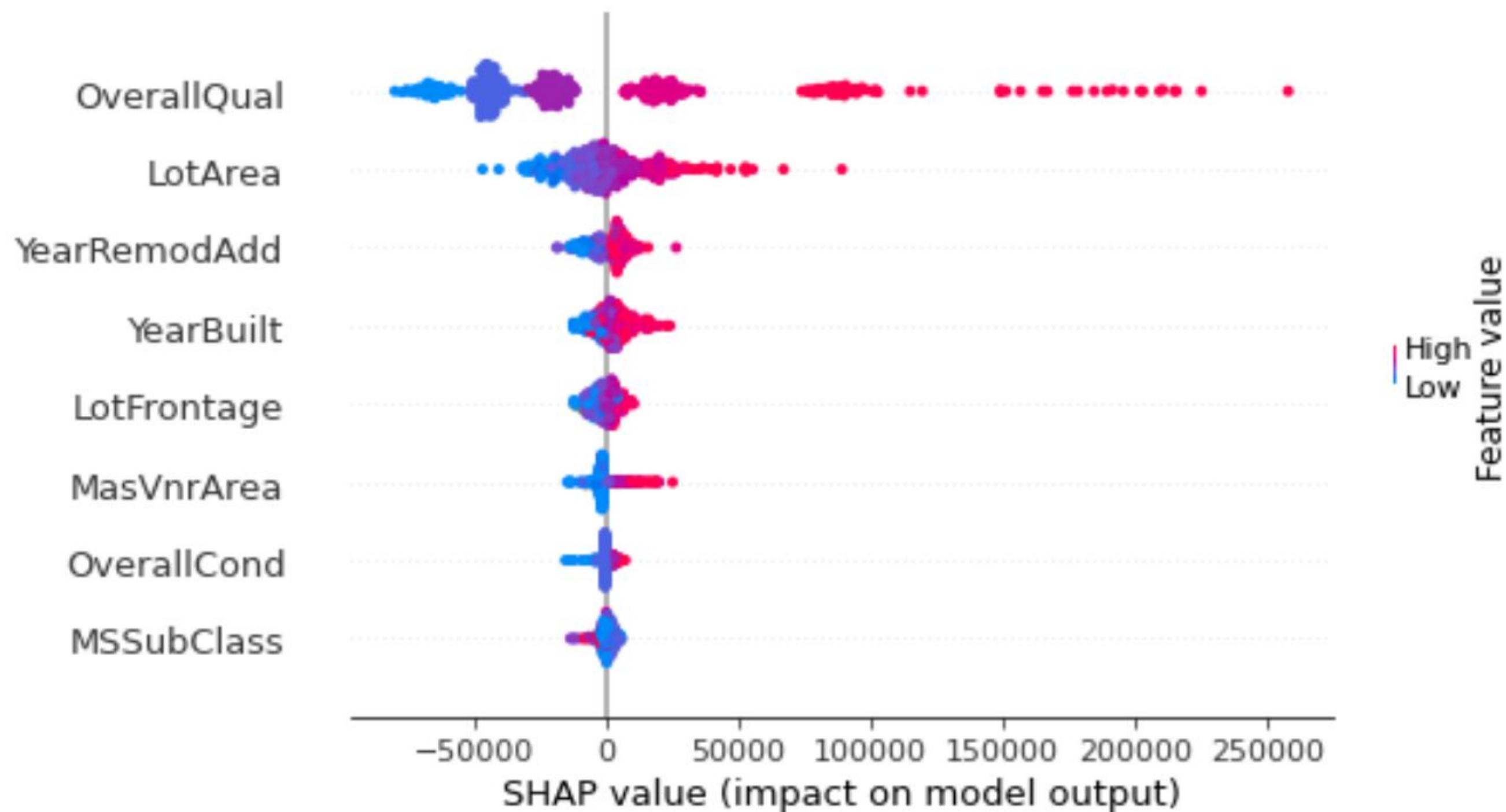
Dependence Plot

```
In [45]: ▶ shap.dependence_plot("YearBuilt", shap_values.values, X_test, interaction_index="LotArea")
```



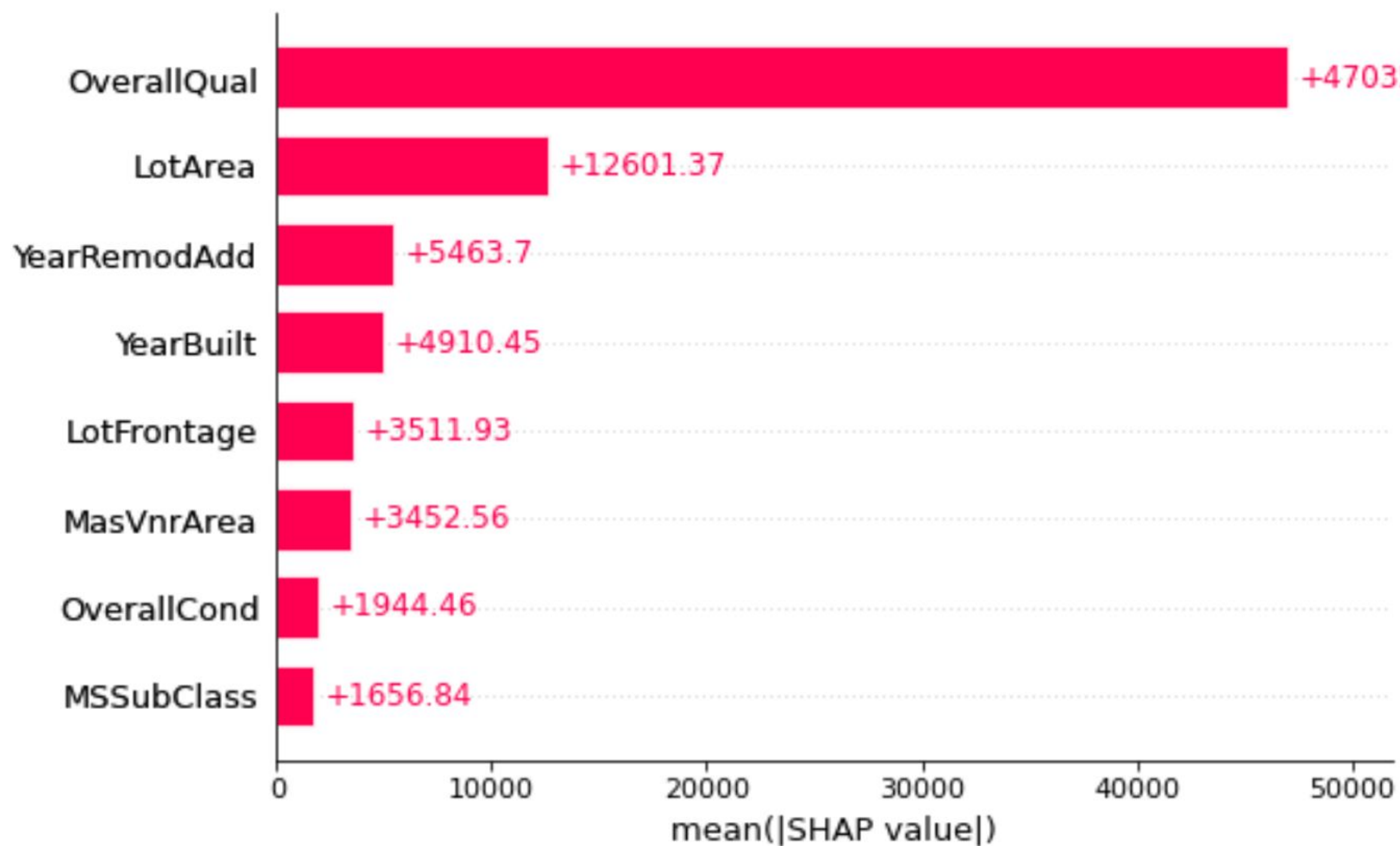
Visão Global

```
In [46]: ▶ # summarize the effects of all the features  
shap.plots.beeswarm(shap_values)
```



Visão Global

```
In [47]: ▶ shap.plots.bar(shap_values)
```



Drawbacks do SHAP

O SHAP é extremamente *lento* para qualquer caso que não seja árvore ou não seja modelo linear. É bom ter isso em mente

Não é um modelo causal; Ele explica que features o modelo está considerando importantes e cabe a você avaliar se essa relação faz sentido (_debugging_ de modelo) e os actionables que são possíveis em cima disso. **Não existe relação entre mudar x unidades do valor da feature e isso impactar em y o valor da saída**

> Correlação não implica Causalidade (sempre bom lembrar)