Construindo um modelo na vida real

1. **Definir o problema:** Compreender claramente qual é o objetivo da modelagem preditiva e o que será feito é essencial. Temos que definir claramente como vai ser a construção dos dados e o que queremos resolver, por exemplo, para prever se um produto em leilão vai ser vendido ou não, não podemos usar uma feature de dias que levou para vender o produto, que seria uma feature comum a ser usada na base de dados de produtos vendidos, não podemos usar ela uma vez que na hora de colocar um novo produto no modelo e determinar se ele vai ser vendido, não teremos a informação de quantos dias levou pra ele ser vendido, até por que ele nem esta a venda ainda.
2. **Coletar os dados:** Obter dados relevantes para o problema que foi definido, focar no que é preciso para determinar a solução do problema.
3. **Separar os dados em treino e teste**.Lembrando que temos que separar os dados antes de fazer pré processamentos e aplicar separadamente no treino e teste, uma vez que pode levar a usarmos dados dos testes para melhorar o treino, fazendo com que o modelo não saia como esperado em dados da vida real.
4. **Preparar os dados**: realizar pré processamento, como limpeza, tratamento de valores missing, normalização e tratamento de variáveis categóricas.
5. **Escolher um modelo e treiná-lo**:
6. **Validar e ajustar o modelo**: A depender do resultado com os dados de teste, deve-se ajustar os hiperparâmetros do modelo em busca de uma melhoria no desempenho dele.
7. **Avaliar o modelo final**:
8. **Implantação e monitoramento**: Implementar um modelo em um ambiente de produção e monitorar seu desempenho ao longo prazo e fazer ajustes com base nos resultados.

Observações importantes

1. **Tuning e feature selection:** Antes de querer tunar o modelo até não dar mais, preocupe-se mais com feature engineer, tentar criar novas features, tirar outras.
2. **Base de dados grande:** Caso a base de dados seja muito grande, tente pegar uma amostra aleatória dela pra escolher o melhor modelo e tomar algumas decisões e depois sumir para a base de dados original.
3. **Scaling:** Algoritmos que utilizam gradiente descendente para calcular os pesos se beneficiam muito da aplicação de Scaling.
4. **Função logaritmo no target:** Transformar o target em Logaritmo: No exemplo da aula, ao transformar o target “aluguel” em logaritmo, houve uma melhora significativa, abaixo tem os possíveis motivos: **Redução de assimetria:** em casos que o target tem valores mais distantes, o log faz com que essa distribuição se aproxima da normal basicamente resultados que indicam uma maior aproximação dos valores dos dados
5. **Métrica que avalia modelo erroneamente**: Nem sempre a acurácia vai ser uma métrica válida para analisar modelos, por exemplo, em casos de churn que temos muitas vezes dados desbalanceados, tendo por exemplo 1% dos casos apenas de churn, se o modelo prever tudo como não chrun, vai ter 99% de acurácia, mas um péssimo modelo, então é importante atentar para qual métrica está sendo usada para avaliar o modelo e se ela é condizente com a realidade.
6. **Pipeline com vários modelos:** Para testar vários modelos no pipeline podemos simplesmente fazer o pipeline ser uma função e chamar um loop passando vários modelos e várias combinações de pré processamento como parâmetro e analisar qual tem melhor performance..

Data Leakage

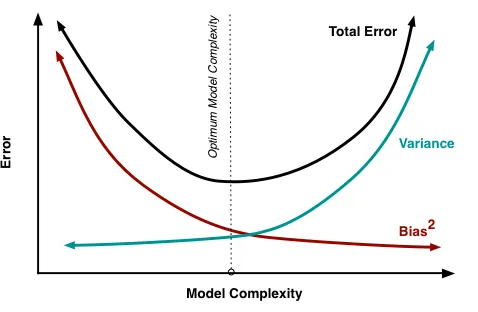
Acontece quando as informações do teste ou validação vazam para o conjunto de treinamento durante o pré-processamento ou modelagem, fazendo com que você tenha um modelo muito bom, mas apenas ilusório, pois seu modelo “roubou” as informações para ter um resultado bom. Para evitar que isso venha a acontecer:

1. Dividir os dados em conjuntos de treinamento, validação e teste ANTES de qualquer pré-processamento:
   1. Não se aplica encoder, input de missing ou scaler antes de divisão treino/teste.
   2. Aplicamos o FIT\_TRANSFORM no treino e apenas TRANSFORM no teste.
2. Evitar usar informações do conjunto de validação ou teste para tomar decisões de modelagem, como ajuste de hiperparâmetros
3. Tomar cuidado com a construção de features e do target, levando em consideração o mesmo exemplo do ebay, não podemos levar em consideração a quantidade de lances que ele teve, por exemplo, pois na hora de lançar um produto não temos essa informação.

Overfitting e underfitting

1. **Viés (Bias):** O viés é a diferença entre a previsão média do nosso modelo e o valor correto que estamos tentando prever.então o *Bias* ou Viés é a distância, em geral, das previsões para o valor correto, Modelos com alto viés são muito simplificados e assumem muitas suposições sobre os dados, o que resulta em um ajuste ruim aos dados de treinamento e teste. Este é um caso típico de underfitting.
2. **Variância:** Variância é basicamente o quanto as previsões do modelo variam em determinado ponto de dados, caso fosse um modelo que tivesse generalizado bem as previsões, existiria uma espécie de homogeneidade das previsões para cada ponto de dados, mas caso a variância seja muito alta em determinado recorte, provavelmente esse modelo overffittou.

Idealmente é ter ambos baixos, porém isso é bem difícil, caso aumentemos a complexidade do modelo, a variância possivelmente vai subir, enquanto ao diminuirmos, o Viés subirá, então é preciso lidar com o trade-off desses dois.



Inferência e Predição

**Inferência**: É um processo matemático que permite que cientistas de dados possam tirar conclusões a partir de conjuntos de dados, entender a relação entre duas features, por exemplo. Esse entendimento é feito em um conjunto menor de dados para ser aplicado em uma população.

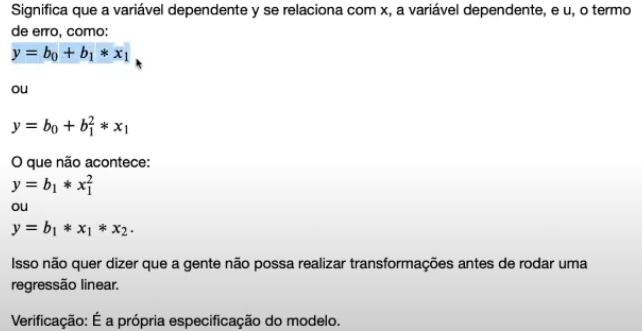
**Predição**: Usar um modelo estatístico ou algoritmo de machine learning para gerar um resultado provável baseado em dados de entrada, diferente da inferência, onde queremos entender a relação entre target e as features, aqui o que importa é simplesmente prever o target.

Existem casos em que só a predição não basta, temos que entender as relações entre as variáveis, por exemplo, para prever a demanda, temos que, além de prever com base nas variáveis, entender exatamente como essa demanda é afetada, para passar futuramente para um gestor, por exemplo, passar apenas a demanda seria muito superficial, pois não saberíamos o que influencia a demanda, saberíamos apenas como prever ela.

A regressão linear, por meio da atribuição de valores às variáveis, faz a inferência, ou seja, acha a relação entre as variáveis. Ou seja, a regressão linear é um método para fazer inferências. No caso, isso acontece quando as escalas das features são parecidas, pois se forem muito distantes, o valor dos coeficientes não vão medir exatamente o nível de importância para prever algo, ou seja a inferência, uma vez que se a escala for muito alta, um coeficiente baixo já vai conseguir fazer uma relação de crescimento do target.

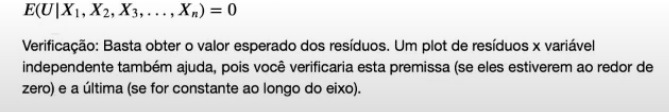
Obtendo Estimadores BLUE(Premissas da regressão Linear)

1. **Linearidade nos parâmetros**



1. **Amostragem aleatória** : Geralmente tomamos como verdade
2. **Média condicional do termo de erro é zero**

O termo de erro é basicamente a diferença entre a predição e o real, ele vai se atualizando a cada iteração



1. **Ausência de multicolinearidade perfeita**

Esse é o mesmo problema da forma normal 4, não podem ter variáveis que se expliquem uma das outras sem ser o target, a variável independente X1 não pode prever X2 e por aí vai, para calcular isso usamos o VIF:

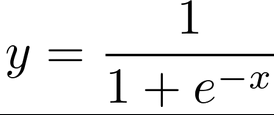


1. **Homoscendasliciedade**

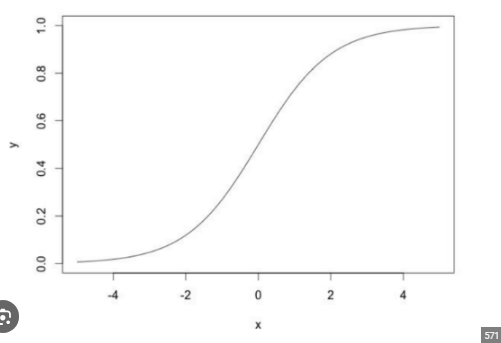
O erro para todas as faixas de valor do target tem que ser parecidas, tanto para o target sendo 1000 quanto para o target sendo 5. Se não acontecer isso, o modelo está enviesado.

Essas 5 premissas garantem que esse é o melhor estimador possível, garantem que ele não terá viés. Quando o trabalho for preditivo, não precisa de tanto vigor para seguir as premissas, mas em trabalhos de inferência sim.

Função Sigmoid



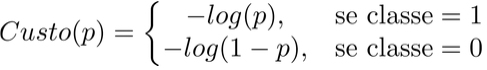
O resultado da função sempre será maior do que zero e menor do que 1: Nos limites de uma probabilidade. Quanto maior o X, mais perto de 1, quanto menor, mais perto de 0.



A função sigmoid é muito utilizada, principalmente nos algoritmos de regressão logística, que é basicamente uma adaptação da regressão linear para problemas de classificação, onde vamos usar primeiramente uma regressão e jogar a saída dessa regressão na função sigmoid, a partir disso vamos ter uma probabilidade dele pertencer a uma das classes:  
 

Basicamente a regressão logística segue o seguinte passado a passo:

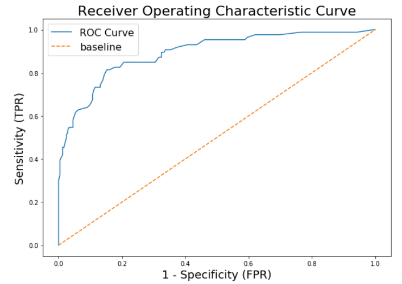
1. Usar uma regressão linear para prever o TARGET
2. Jogar a saída a regressão na função sigmoid
3. Analisar a que classe ela pertence com base no sistema acima

Funções perda geralmente utilizada para o treinamento de regressão logística:  


Sendo que esse P é a probabilidade, ou seja, o output da função sigmoid.

Curva ROC e ROC AUC

A curva mostra a proporção da taxa de verdadeiro positivo(vertical) e taxa de falso positivo(horizontal) em diferentes faixas de probabilidades escolhidas.



Sendo a fórmula para esses dois:

 Isso é o recall



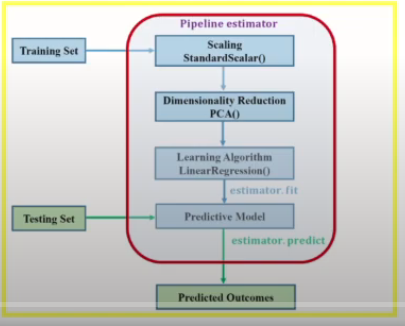
* TPR: Proporção de positivos reais que foram corretamente identificados como negativo
* FPR: Proporção de negativos reais que foram incorretamente identificados como positivos

A medida que diminui o recall, diminuindo a confiança para ser da classe 1, por exemplo, realmente vão diminuir os FN, mas vão aumentar os registros que não eram da classe 1, mas foram erroneamente para ela, ou seja, os FP, que é a métrica medida pelo FPR, ou seja, um tradeoff.

Temos que nos atentar a curva ROC para não tentarmos apenas diminuir o recall despretensiosamente sem pensar nas consequências geradas, que geralmente é o aumento do FP.

Pipeline

Organiza as etapas do tratamento de dados e de treinamento, organizando melhor o código



Outro modo de criar PipeLine(quando todas as colunas são numeradas, evitando o uso do column transform)

