Data Science para Negócios III

Visualização e Storytelling de Dados

Prof. Klayton R. Castro

IDP
Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa

11 de abril de 2024

Jupyter Notebooks

- Os Datasets e Jupyter Notebooks utilizados durante o curso estão disponíveis no GitHub para referência, revisão ou prática adicional.
- Visite o repositório `idp-storytelling` na seguinte URL:

`https://github.com/klaytoncastro`

Entendendo as Tarefas de Classificação e Regressão

- Baixe os notebooks no GitHub do Professor com os exemplos de fluxo de trabalho para resolver problemas de **Classificação** e **Regressão**.
- Lembre-se que este curso tem o objetivo de esclarecer os fundamentos do aprendizado de máquina. Para isso, analise o código calmamente, executando os notebooks **célula por célula**. Evite simplemente copiar e colar código sem entendê-lo.
- Agora carregue os notebooks em seu ambiente `Jupyter` ou `Google Colab` e analise os fluxos de trabalho que adotamos para criação de uma máquina preditiva em um contexto de aprendizado supervisionado.

Fluxo de trabalho para o Aprendizado de Máquina

Observe como utilizamos as bibliotecas `Pandas` e `Numpy` para manipulação de dados, `Seaborn` e `Matplotlib` para visualização e `Scikit-Learn` para carregar os algoritmos empregados para criar os dois modelos de Machine Learning. De maneira resumida, programamos uma máquina preditiva capaz de:

- Executar a tarefa de classificação do tipo de vinho (branco ou tinto);
- Executar a tarefa de regressão para a previsão de sua qualidade (nota).

Análise Exploratória de Dados

- Antes de abordarmos a modelagem preditiva e identificarmos eventual margem de otimização, é importante enfatizar a estatística descritiva como base para obter as características de cada variável e observar o impacto dessas características (*features*) em nosso conjunto de dados (*dataset*).
- Isso inclui calcular medidas de tendência central (média, mediana), dispersão (desvio padrão, intervalo interquartil), além de explorar a distribuição de cada variável (contagem, valores únicos, possíveis outliers), bem como a relação entre as variáveis preditoras e a variável alvo.

Análise Exploratória de Dados

- Primeiramente carregamos o *Dataset* em formato `CSV` e realizamos as operaçoes e manipulação dos dados necessárias.
- Depois disso, realizamos a análise de distribuição de cada variável numérica usando histogramas.
- Efetuamos também a análise de Boxplots para identificar *outliers*, ou seja, pontos que se destacam significativamente dos demais em um conjunto de dados, estando distantes da maioria das demais observações da amostra.

Análise Exploratória de Dados

- Visualização Geral: obtivemos uma visão geral do dataset através dos métodos ‘.describe()‘ e ‘.info()‘ para uma visão geral do tipo de dados e verificação de valores ausentes (missing values).
- Análise Descritiva: obtivemos as medidas de tendência central e dispersão para cada variável.
- Realizamos a contagem de valores para a variável categórica (color) e discreta (quality), alvos de nosso modelo de Machine Learning.

Escolha do Algoritmo

- Depois de analisar o código para criar nosso Primeiro Modelo de Aprendizado de Máquina, optamos pelo robusto algoritmo *ExtraTrees* para criar nosso primeiro modelo de Machine Learning.
- Este algoritmo utiliza o conceito de adotar múltiplas árvores de decisão para realizar as tarefas de classificar os vinhos em tintos ou brancos e, em seguida, usamos a mesma técnica para predizer a qualidade (nota) conforme análise de suas propriedades químicas.

Modelo de Aprendizado de Máquina

Pronto! Agora que compreendemos o fluxo de trabalho para criação do modelo, vimos a importância da análise exploratória para entendimento dos dados e executamos tarefas comuns de pré-processamento, seleção de dados de treino e teste, escolha do algoritmo e modelagem básica para criação de uma máquina preditiva funcional.

Tarefa 01

Após executar os notebooks **passo a passo**, **entender** o que o código está realizando e efetuar os **ajustes** necessários, responda:

a. Quais células precisam ser ajustadas no notebook da tarefa de classificação? Por que?

b. Quais células precisam ser ajustadas no notebook da tarefa de regressão? Por que?

c. Qual variável aparece com mais outliers? Como isso pode interferir no nosso modelo de ML?

Classificação com outros algoritmos

Para problemas de classificação, além do algoritmo *ExtraTreesClassifier*, os algoritmos *Naive Bayes (NB)* e *Support Vector Machine (SVM)* são alternativas populares, dependendo da natureza dos dados e do problema específico que você está tentando resolver.

- NB é uma técnica de classificação baseada na ideia de aplicar o teorema de Bayes com a "ingenuidade"de supor independência entre os preditores.
- É fácil de construir e particularmente útil para grandes volumes de dados.
- Costuma ser eficaz em problemas de classificação multinomial e binomial.

Usando Naive Bayes

Há diferentes implementações de NB no Scikit-Learn, adequados para diferentes tipos de dados.

- GaussianNB: Adequado para classificação de features contínuas que seguem uma distribuição normal.
- BernoulliNB: Adequado para features binárias.
- MultinomialNB: Adequado para features que são contagens ou frequências de termos (comumente usado em classificação de texto).

Código Python

Experimente a implementação do algoritmo NB e avalie os resultados em comparação à Árvore de Decisão com ExtraTreesClassifier. Use as métricas de desempenho acurácia, precisão, recall e F1-score para isso. Segue exemplo de código:

```
1 from sklearn.naive_bayes import BernoulliNB
2
3 modelo = BernoulliNB()
4 modelo.fit (x_train, y_train)
5
6 y_pred = modelo.predict(x_test)
```

Usando SVM

O Support Vector Machine (SVM) é um método poderoso e versátil para tarefas de classificação e detecção de outliers. Para classificação, especialmente em casos de categorias claramente distintas, o algoritmo SVM pode ser eficaz.

- O Scikit-Learn oferece várias implementações do SVM, incluindo SVC (Support Vector Classification), que é comumente usado para problemas de classificação.
- Teste e avalie os resultados usando as métricas apropriadas para classificação. Consulte a documentação do Scikit-learn para obter maiores detalhes sobre os *kernels*.

Código Python

```
1 from sklearn.svm import SVC
2 # Inicializando o classificador SVM com um
    kernel. O kernel 'rbf' pode ser alterado para
     'linear', 'poly', etc.
3
4 modelo_svm = SVC(kernel='linear')
5 modelo_svm.fit(X_train, y_train)
6 y_pred = modelo_svm.predict(X_test)
```

Usando Regressão Logística

Embora seja chamada de regressão, esta técnica é utilizada para classificação binária (prever entre duas classes).

- Estima probabilidades usando uma função logística que mapeia qualquer valor real para um valor entre 0 e 1.
- É ideal para problemas onde a variável dependente é categórica (por exemplo, sim/não, verdadeiro/falso).

Código Python

```
1 from sklearn.linear_model import
      LogisticRegression
2 modelo = LogisticRegression()
3 modelo.fit(X_train, y_train)
```

Avaliação

Após testar os algoritmos, você precisa desenvolver uma abordagem e avaliar quão bem seu modelo irá performar. Para esta tarefa, seguem as métricas mais usuais:

- **Acurácia**: Proporção de previsões corretas (positivas ou negativas) em relação ao total de casos analisados. Mede a eficácia geral do modelo.
- **Precisão**: Proporção de previsões corretas positivas (VP) em relação ao total de previsões positivas feitas (VP + FP). Indica a exatidão das previsões positivas.
- **Recall**: Proporção de previsões corretas positivas (VP) em relação ao total de casos positivos reais (VP + FN). Mede a capacidade do modelo de identificar todos os casos relevantes.
- **F1-score**: Média harmônica entre precisão e recall. É útil quando se deseja um equilíbrio entre precisão e recall, especialmente se as classes estiverem desbalanceadas.

Tarefa 2

Após **experimentar** os algoritmos acima e **compreender** sua implementação, gere as **Curvas ROC** para cada um deles e responda:

- Qual algoritmo performou melhor em termos de **métricas**?
- Qual algoritmo performou melhor em termos de **velocidade**?
- Explique os comportamentos das curvas ROC para cada algoritmo.

Regressão com outros algoritmos

Para a tarefa de regressão, no caso, prever um número inteiro que reflete a qualidade do vinho, existem vários algoritmos disponíveis no Scikit-Learn.

- Inicialmente utilizamos uma abordagem de **Árvore de Decisão** com a implementação *ExtraTreesRegressor*.
- Dentre outras alternativas populares destacamos: **SVR**, **Regressão** (*Linear, Ridge, Lasso*) e **Random Forest Regressor**.

Support Vector Machine (SVM) para Regressão (SVR)

- O Support Vector Regression (SVR) é a versão do SVM que pode ser usada para problemas de regressão.
- O SVR pode ser eficaz em espaços de alta dimensionalidade, até mesmo em casos onde o número de dimensões excede o número de amostras.

```
1  from sklearn.svm import SVC
2  # Inicializando o classificador SVM com
       um kernel. Voce pode experimentar com
        diferentes kernels como 'linear', '
       poly', 'rbf', etc.
3  from sklearn.svm import SVR
4  modelo_svr = SVR(kernel='linear')
5  modelo_svr.fit(x_train, y_train)
6  y_pred = modelo_svr.predict(x_test)
```

Regressão Linear

A regressão linear é um dos métodos mais simples, mas amplamente utilizado.

- Costuma ser um bom ponto de partida para problemas de regressão, sendo eficaz para estimar relações lineares diretas entre variáveis dependentes e independentes.
- Sensível a outliers e a multicolinearidade, um fenômeno no qual duas ou mais variáveis preditoras são altamente correlacionadas, o que pode causar problemas na estimação dos coeficientes do modelo de regressão.

```
1 modelo_lr.fit(x_train, y_train)
2 y_pred = modelo_lr.predict(x_test)
```

Otimizando a Modelagem

No contexto de modelagem estatística e machine learning, viés e variância são dois componentes fundamentais do erro de previsão de qualquer modelo:

- Um modelo com alto viés não aprende bem os detalhes e padrões nos dados (underfitting), simplificando em demasia um problema complexo.
- Um modelo com alta variância é fortemente influenciado por detalhes mínimos dos dados de treinamento, o que pode levar a modelagem do ruído dos dados ao invés dos padrões reais (overfitting).
- Ou seja, um modelo simples demais aumenta o viés e reduz a variância (risco de underfitting), enquanto um modelo complexo demais reduz o viés e aumenta a variância (risco de overfitting).
- Para evitar isso, podemos adotar métodos de regularização como *Ridge* e *Lasso*, capazes de ajustar o compromisso viés-variância adicionando um termo de penalidade ao modelo.

Regressão Ridge

Adiciona uma penalidade igual ao quadrado dos coeficientes multiplicados por um fator α. Isso reduz a variância, pois impede que os coeficientes atinjam valores muito altos, mas aumenta o viés, pois o modelo é forçado a ser menos sensível aos dados.

```
1
2       from sklearn.linear_model import Ridge
3       modelo_ridge = Ridge(alpha=1.0) # O
            parametro alpha controla a forca da
            regularizacao.
```

Regressão Lasso

Adiciona uma penalidade igual ao valor absoluto dos coeficientes multiplicados por α. Além de ajustar o viés e a variância, pode reduzir o número de variáveis no modelo (*zeroing out coefficients*), uma forma de seleção automática das features.

```
1
2     from sklearn.linear_model import Lasso
3     modelo_lasso = Lasso(alpha=0.1)
```

Árvores de Decisão Aleatórias

São extensões otimizadas das árvores de decisão, cuja implementação combina múltiplas árvores para definir um modelo estatiscamente mais robusto (método ensemble).

- Em uma Random Forest, muitas árvores de decisão são criadas usando um subconjunto aleatório dos atributos (features), ao invés de utilizar todas as variáveis disponíveis.
- Esta técnica é conhecida como "feature bagging"e ajuda a descorrelacionar as árvores, aumentando a diversidade do modelo.
- Cada árvore individual faz sua própria previsão de valor, e a previsão final é tipicamente a média (ou a mediana) das previsões de todas as árvores.
- Este método de agregação é conhecido como averaging e ajuda a reduzir a variância sem aumentar o viés.

Comparando implementações

Os algoritmos *Random Forest Regressor* e *ExtraTrees Regressor* (*Extremely Randomized Trees*), que utilizamos inicialmente, possuem diferenças significativas na forma de construir a floresta e dividir os dados, o que pode afetar seu desempenho e aplicabilidade.

- O Random Forest Regressor constrói a floresta de forma relativamente conservadora. Utiliza amostragem com reposição (bootstrap sampling) para selecionar os dados e, em cada divisão, seleciona aleatoriamente um subconjunto de características (features) para compor cada árvore.
- Esta técnica visa garantir que as árvores sejam distintas, reduzindo a variância do modelo sem aumentar muito o viés, o que previne o overfitting.

Comparando implementações

Já o *ExtraTreesRegressor* leva a ideia de aleatoriedade um passo adiante para aumentar ainda mais a diversificação entre as árvores.

- Embora adote bootstrap sampling, ao contrário do Random Forest, que busca a melhor divisão em cada subconjunto de dados, o ExtraTrees faz divisões escolhendo pontos de corte totalmente ao acaso para cada feature e usa a melhor dessas divisões aleatórias.
- A ideia é que esta aleatoriedade extra ajude a criar modelos que generalizam melhor. Ou seja, oferece uma redução na variância mas, potencialmente, a um aumento no viés.
- Entretanto, costuma ser mais rápido para treinar, já que não precisa encontrar a melhor divisão possível para cada subconjunto de features, mas pode ser menos sensível a outliers e ter sua precisão reduzida em certos conjuntos de treinamento.

Usando Random Forest Regressor

```
1  from sklearn.ensemble import
       RandomForestRegressor
2  modelo_rfr = RandomForestRegressor()
```

Avaliação

Após experimentar os algoritmos, você avaliar como eles estão performando. Seguem as métricas de avaliação usuais em tarefas de regressão:

- **MAE (Mean Absolute Error)**: Média dos valores absolutos dos erros entre previsões e valores reais. Oferece uma ideia da magnitude média dos erros.
- **MSE (Mean Squared Error)**: Média dos quadrados dos erros. Penaliza mais erros maiores, sendo útil quando grandes erros são indesejáveis.
- **RMSE (Root Mean Squared Error)**: Raiz quadrada do MSE. Retorna a taxa à mesma unidade das variáveis de interesse e mais é sensível a grandes desvios, sendo comumente usada por sua interpretabilidade.
- **R^2 (Coefficient of Determination)**: Proporção da variância dos dados que é explicada pelo modelo. Valores próximos de 1 indicam que o modelo explica uma grande parte da variabilidade dos dados.

Tarefa 03

Após **experimentar** os algoritmos acima e **compreender** sua implementação, responda:

- Qual algoritmo performou melhor em termos de **métricas**?
- Qual algoritmo performou melhor em termos de **velocidade**?

Tarefa 04

Agora que você testou várias alternativas para tratar problemas de classificação, regressão e compreendeu as métricas para avaliar o desempenho nestes tipos de tarefas, vamos resolver o desafio de predizer a qualidade dos vinhos de uma forma diferente:

- Selecione uma nota de corte para definir se um vinho é bom ou ruim em termos de qualidade. Por exemplo, considerando o histograma do atributo quality, vimos que os melhores vinhos da nossa amostra possuem nota $>= 7$.
- Transforme os dados desta coluna ou crie uma nova coluna no dataframe para reduzir o problema de regressão a um problema de classificação, atribuindo valores binários para classificar os vinhos.
- Compare os resultados entre as abordagens: no primeiro momento com regressão e no segundo como classificação, utilizando a nota de corte definida para predizer se o vinho é um dos melhores ou não.

Colocando o modelo em produção

Agora que o modelo está treinado e desempenhando como MVP (*Minimum Viable Product*), como transformar um programa de machine learning em uma aplicação em produção?

> *MVP é um conceito usado no desenvolvimento de produtos em startups e projetos de inovação tecnológica para descrever uma versão simplificada de um novo produto que possui apenas as funcionalidades essenciais para ser lançado.*