Data Science para Negócios III

Visualização e Storytelling de Dados

Prof. Klayton R. Castro

IDP
Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa

5 de abril de 2024

Entendendo as Tarefas de Classificação e Regressão

- Baixe os notebooks com os exemplos de código para Classificação e Regressão.
- Utilizamos as bibliotecas Pandas e Numpy para manipulação dos dados, Seaborn/Matplotlib para visualização e Scikit-Learn para criar os modelos de Machine Learning.
- Execute os notebooks, passo a passo, em seu ambiente Jupyter ou Google Colab para compreender como abordamos inicialmente o fluxo de trabalho para criação de uma máquina preditiva capaz de executar as tarefas de classificação de vinhos em brancos ou tintos e a previsão de sua qualidade (nota).

Análise Exploratória de Dados

- Antes de abordarmos a otimização da modelagem preditiva de aprendizado de máquina, vamos enfatizar a estatística descritiva para obter as características de cada variável e observar o impacto dessas 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, realizamos as operaçoes e manipulação dos dados necessárias.
- Depois disso, realizamos a análise de distribuição de cada variável numérica usando histogramas.
- Realizamos também a análise de Boxplots para identificar outliers.
- Qual variável aparece com mais outliers? Como isso pode interferir no nosso modelo de ML?

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.

Tarefa 01

- Nosso primeiro objetivo era prever a cor do vinho (branco ou tinto), uma tarefa de classificação.
- Em seguida, o objetivo era prever a qualidade do vinho. Em princípio, uma tarefa de regressão.
- Para isso adotamos o robusto algoritimo **ExtraTrees** para criar nosso primeiro modelo de Machine Learning, utilizando o conceito de árvore de decisão para as tarefas de classificar os vinhos em tintos ou brancos e, em seguida, para predizer a qualidade (nota) conforme análise de suas propriedades químicas.

Nosso Primeiro Modelo de Aprendizado de Máquina

Após executar os notebooks passo a passo e entender o que o código está realizando, 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?

Tarefa 02

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

- Naive Bayes é uma técnica de classificação baseada em 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. Além disso, é eficaz em problemas de classificação multinomial e binomial.

Usando Naive Bayes

Existem diferentes implementações de Naive Bayes no Scikit-Learn, adequados para diferentes tipos de dados.

- GaussianNB: Usado em classificação onde as features são contínuas e seguem uma distribuição normal.
- BernoulliNB: Adequado para features binárias.
- MultinomialNB: Bom para quando suas features são contagens ou frequências de termos (comumente usado em classificação de texto).

Código Python

Teste as implementações sugeridas no slide anterior e avalie os resultados. Use as métricas de desempenho acurácia, precisão, recall e F1-score para isso. Segue exemplo de código:

```
from sklearn.naive_bayes import GaussianNB

# Para dados com features continuas que seguem uma
    distribui  o aproximadamente normal
modelo_nb = GaussianNB()

modelo_nb.fit(X_train, y_train)
y_pred = modelo_nb.predict(X_test)
```

Usando SVM

O Scikit-Learn oferece várias implementações do SVM, incluindo SVC (Support Vector Classification), que é comumente usado para problemas de classificação.

- 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.
- Teste e avalie os resultados usando as métricas apropriadas para classificação. Consulte a documentação do Scikit-learn sobre os kernels.

Código Python

```
from sklearn.svm import SVC
# Inicializando o classificador SVM com um kernel. O
    kernel 'rbf' pode ser alterado para 'linear', 'poly
    ', etc.

modelo_svm = SVC(kernel='linear')
modelo_svm.fit(X_train, y_train)
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

```
from sklearn.linear_model import LogisticRegression

# Criando uma inst ncia do modelo
model = LogisticRegression()

# Treinando o modelo
model.fit(X_train, y_train)
```

Avaliação dos Modelos

Após testar os algoritmos, você precisa desenvolver uma abordagem e avaliar quão bem seu modelo irá performar. Para uma tarefa de classificação, vimos que métricas comuns incluem:

- **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.

Código Python

O Scikit-Learn fornece funções prontas para calcular essas métricas.

```
from sklearn.metrics import accuracy_score,
    precision_score, recall_score, f1_score

print("Acuracia:", accuracy_score(y_test, y_pred))
print("Precisao:", precision_score(y_test, y_pred,
    average='macro'))
print("Recall:", recall_score(y_test, y_pred, average=
    'macro'))
print("F1-score:", f1_score(y_test, y_pred, average='
    macro'))
```