Aprendizado de Máquina - Atividade Prática Naïve Bayes

Willian Reichert (134090)

Análise do questionário

a) Verdadeiro, visto que essa é a ideia do algoritmo: a probabilidade condicional da ocorrência do valor de um atributo dada a ocorrência de uma classe específica leva em consideração apenas a quantidade de vezes que esse valor aparece dentre os dados pertencentes a essa classe.

b) Verdadeiro. Como $P(target = acc) = 0.4$ e $P(target = unacc) = 0.6$, a probabilidade de uma instância ser da classe *unacc* é maior. Esses valores podem ser calculados rapidamente pela divisão da quantidade de instâncias da cada classe pela quantidade total de instâncias.

c) Verdadeiro. O cálculo é feito pela divisão da quantidade de instâncias de *acc* que possuem o valor *med* no atributo *price* pela quantidade de instâncias de *acc*, o que nos dá 3/10.

d) Parcialmente falso. A instância será classificada pelo modelo como *acc*, porém as probabilidades a posteriori de ambas as classes são diferentes: $P(target = acc|x) = 0.0289$ e $P(target = unacc|x) = 0.0249$. A implementação do algoritmo para esse dataset específico foi realizada em Python, e o código está no final desse relatório.

e) Verdadeiro. Como não foi utilizada a correção de Laplace, a probabilidade $P(safety = low|acc)$ é igual a zero, já que não há nenhuma instância de *acc* com o atributo *safety* igual a *low*.

Código

O código abaixo, implementado para a resolução dos três itens que envolvem cálculos, foi desenvolvido em Python 3.10 com a utilização da biblioteca Pandas. Ele também foi enviado pelo Moodle em um arquivo compactado juntamente com esse relatório.

```python
from pandas import read_csv, DataFrame


def main() -> None:
    # leitura dos dados da base para um dataframe
    data: DataFrame = read_csv('data.csv')

    # item b do questionario
    p_acc: float = data['target'].value_counts()['acc'] / data.shape[0]  # P(target = acc)
    p_unacc: float = data['target'].value_counts()['unacc'] / data.shape[0]  # P(target = unacc)
    print(f'P(target = acc) = {p_acc}')
    print(f'P(target = unacc) = {p_unacc}')
    print(f'P(target = unacc) > P(target = acc)? {p_unacc > p_acc}', end='\n\n')

    # item c do questionario
    acc_data: DataFrame = data.query('target == "acc"')  # filtra os dados da classe acc
    pcond_price_med: float = acc_data['price'].value_counts()['med'] / acc_data.shape[0]  # P(price = med | target = acc)
    print(f'P(price = med | target = acc) = {pcond_price_med}')
    print(f'P(price = med | target = acc) == 0.3? {pcond_price_med == 0.3}', end='\n\n')

    # item d do questionario
    acc_prod: float = 1.0  # produtorio de P(target = acc | x)
    unacc_prod: float = 1.0  # produtorio de P(target = unacc | x)
    unacc_data: DataFrame = data.query('target == "unacc"')  # filtra os dados da classe unacc
    test_instance: dict = {'price': 'low', 'lug_boot': 'small', 'safety': 'high'}
    # instancia de teste x
    for attr, value in test_instance.items():
        acc_prod *= acc_data[attr].value_counts()[value] / acc_data.shape[0]  # prod * P(attr = value | target = acc)
```

```
        unacc_prod *= unacc_data[attr].value_counts()[value] / unacc_data.shape[0]  # prod * P(attr = value | target = unacc)
    ppost_acc: float = p_acc * acc_prod  # P(target = acc | x)
    ppost_unacc: float = p_unacc * unacc_prod  # P(target = unacc | x)
    print(f'P(target = acc | x) = {ppost_acc:.5f}')
    print(f'P(target = unacc | x) = {ppost_unacc:.5f}')
    print(f'P(target = acc | x) > P(target = unacc | x)? {ppost_acc > ppost_unacc}')
    return


if __name__ == '__main__':
    main()
```