Relatório trabalho prático

Aprendizagem Automática

Trabalho realizado por:
Ruben Farinha nº48329
Gustavo Gomes nº48392

Docentes:
Professora Teresa Gonçalves

1. Introdução

O relatório descreve o desenvolvimento e a análise de duas classes, KNeighborsClassUE e NBayesClassUE, destinadas à implementação dos algoritmos KNN (K-Nearest Neighbors) e Naïve Bayes para problemas de classificação com atributos nominais, num ambiente scikit-learn, ou seja, os parâmetros de entrada e saída dos métodos deverão permitir a substituição destes por outros algoritmos de classificação implementados no scikit-learn.

2. Main

Nos códigos-fonte dos algoritmos, foi desenvolvida uma classe main dedicada à execução de testes das classes KNeighborsClassUE e NBayesClassUE. O processo inicia-se com a escolha do conjunto de treino a ser utilizado, com a leitura do arquivo correspondente. Em seguida, são separadas as classes dos atributos, e os conjuntos de treino e teste são criados através da função train_test_split.
Após a preparação dos dados, procedemos à definição dos valores de k e p para o KNN, e alpha para o Naïve Bayes. A execução dos testes nos conjuntos de dados, utilizando diferentes combinações de parâmetros, possibilita a obtenção de métricas como a precisão e exatidão que nos permitirá fazer uma análise critica do desempenho dos algoritmos.

3. Métodos

Class KNeighborsClassUE:

__init__(self, k=3, p=2.0): Este método é o construtor da classe KNeighborsClassUE, inicializando os parâmetros k (número de vizinhos mais próximos) e p (potência para a métrica de Minkowski). Além disso, são inicializados os atributos X_train e y_train que armazenam os dados de treino durante o processo de ajuste do modelo.

fit(self, X, y): O método fit é responsável por treinar o modelo KNN com os dados de treino fornecidos. Ele armazena os conjuntos de treino X_train e y_train na instância da classe para posterior referência.

predict(self, X): O método predict recebe um conjunto de dados de teste X e retorna as previsões para cada instância no formato de um array. Ele utiliza o método _predict para calcular a classe prevista para cada ponto de teste.

_predict(self, x): Este método auxiliar calcula a classe mais comum entre os k-vizinhos mais próximos para um ponto de teste específico, utilizando a métrica de Minkowski para calcular as distâncias.

_distance(self, x1, x2): O método auxiliar _distance calcula a distância de Minkowski entre dois pontos x1 e x2.

score(self, X, y): O método score avalia o desempenho do modelo, calculando e retornando a exatidão e a precisão para um conjunto de teste fornecido.

Class NBayesClassUE:

init(self, alpha=1): Este é o construtor da classe NBayesClassUE. Ele inicializa o parâmetro alpha do estimador Lidstone, que é utilizado para a suavização na estimativa de probabilidades. Além disso, são inicializados os atributos class_probs (um dicionário para armazenar as probabilidades das classes), feature_probs (um dicionário para armazenar as probabilidades condicionais dos atributos) e classes (uma lista para armazenar as classes presentes nos dados).

fit(self, X, y): O método fit é responsável por treinar o modelo Naïve Bayes com os dados de treino fornecidos. Ele calcula as probabilidades das classes e as probabilidades condicionais das características dadas as classes. O resultado é armazenado nos atributos class_probs e feature_probs.

predict(self, X): O método predict recebe um conjunto de dados de teste X e retorna as previsões para cada instância no formato de um array. Utilizando o método _predict, ele calcula a classe prevista para cada ponto de teste.

_predict(self, sample): Este método auxiliar calcula a classe mais provável para uma amostra específica. Ele percorre todas as classes, calculando a probabilidade logarítmica condicional para cada característica dada a classe. A classe mais provável é aquela que maximiza a soma dessas probabilidades logarítmicas.

score(self, X, y): O método score avalia o desempenho do modelo, calculando e retornando a exatidão e a precisão para um conjunto de teste fornecido. Ele utiliza o método predict para gerar previsões e, em seguida, compara essas previsões com as verdadeiras classes para calcular as métricas de desempenho.

4. Decisões

Class NBayesClassUE:

Relativamente há existência nos dados de teste valores dos atributos e classe que não ocorrem nos dados de treino foi decidido a seguinte implementação, quando um valor não faz parte da lista de valores possíveis para uma dada característica no conjunto de treino, o número de ocorrências\casos para esse valor será igual a zero. Deste modo como não existe este valor no conjunto de treino, não existe probabilidade para a mesma, logo quando for efetuado o cálculo da previsão só serão utilizadas as probabilidades de valores que constam no conjunto de treino.
Decidimos também por optar por guardar as probabilidades condicionais e das classes em dicionários do python para facilitar a sua manipulação e acesso.

Class NBayesClassUE:

Relativamente à existência nos dados de teste de valores dos atributos e classe que não ocorrem nos dados de treino, foi decidido implementar a seguinte abordagem: quando um valor não faz parte da lista de valores possíveis para um dado atributo no conjunto de treino, o número de ocorrências para esse valor será igual a zero. Deste modo, como não existe este valor no conjunto de treino, não existe probabilidade para o mesmo. Portanto, ao calcular a previsão, só serão utilizadas as probabilidades de valores que constam no conjunto de treino. Para facilitar a manipulação e acesso, as probabilidades condicionais e das classes são armazenadas em dicionários do Python.

5. Análise do desempenho

Iris.csv:

Desempenho do modelo no conjunto de treino (k=1, p=1):
- Exatidão: 1.0
- Precisão: 1.0

Desempenho do modelo no conjunto de teste (k=1, p=1):
- Exatidão: 0.9473684210526315
- Precisão: 0.9473684210526315

Desempenho do modelo no conjunto de treino (k=1, p=2):
- Exatidão: 1.0
- Precisão: 1.0

Desempenho do modelo no conjunto de teste (k=1, p=2):
- Exatidão: 0.9473684210526315
- Precisão: 0.9473684210526315

Desempenho do modelo no conjunto de treino (k=3, p=1):
- Exatidão: 0.9642857142857143
- Precisão: 0.9642857142857143

Desempenho do modelo no conjunto de teste (k=3, p=1):
- Exatidão: 0.9473684210526315
- Precisão: 0.9554655870445344

Desempenho do modelo no conjunto de treino (k=3, p=2):
- Exatidão: 0.9642857142857143
- Precisão: 0.9642857142857143

Desempenho do modelo no conjunto de teste (k=3, p=2):
- Exatidão: 0.9473684210526315
- Precisão: 0.9554655870445344

Desempenho do modelo no conjunto de treino (k=5, p=1):
- Exatidão: 0.9732142857142857
- Precisão: 0.9734194015444014

Desempenho do modelo no conjunto de teste (k=5, p=1):
- Exatidão: 0.9473684210526315
- Precisão: 0.9554655870445344

Desempenho do modelo no conjunto de treino (k=5, p=2):
- Exatidão: 0.9732142857142857
- Precisão: 0.9734194015444014

Desempenho do modelo no conjunto de teste (k=5, p=2):
- Exatidão: 0.9473684210526315
- Precisão: 0.9554655870445344

Desempenho do modelo no conjunto de treino (k=9, p=1):
- Exatidão: 0.9732142857142857
- Precisão: 0.9734371987661461

Desempenho do modelo no conjunto de teste (k=9, p=1):
- Exatidão: 0.9736842105263158
- Precisão: 0.975877192982456

Desempenho do modelo no conjunto de treino (k=9, p=2):

- Exatidão: 0.9732142857142857
- Precisão: 0.9734371987661461

Desempenho do modelo no conjunto de teste (k=9, p=2):
- Exatidão: 0.9736842105263158
- Precisão: 0.975877192982456

Os resultados obtidos revelam uma notável consistência no desempenho do modelo, destacando-se pela exatidão e precisão significativamente elevadas em diversas configurações. Este padrão indica uma sólida capacidade do modelo em generalizar para diferentes conjuntos de dados.
Entretanto, ao analisar a configuração com k=1, observa-se uma disparidade entre os resultados no conjunto de treino e teste. A exatidão e precisão perfeitas no conjunto de treino sugerem um possível overfitting, indicando que o modelo pode estar a ajustar-se demasiado aos detalhes específicos do conjunto de treino, o que resulta num desempenho inferior ao generalizar para novos dados.
Por outro lado, a configuração com k=9 demonstra um equilíbrio mais consistente entre o desempenho nos conjuntos de treino e teste. Ambos os conjuntos apresentam resultados elevados, indicando uma capacidade mais robusta do modelo em lidar com diferentes instâncias. Essa configuração parece ser mais adequada para evitar o overfitting, mantendo um desempenho sólido tanto no treino quanto no teste.

bc-nominal.csv:

Desempenho do modelo no conjunto de treino (alpha=0):
- Exatidão: 0.7536231884057971
- Precisão: 0.7476398653627835

Desempenho do modelo no conjunto de teste (alpha=0):
- Exatidão: 0.8285714285714286
- Precisão: 0.8376407166640241

No conjunto de treino, o modelo atinge uma precisão e exatidão razoáveis.
No conjunto de teste, a precisão é ligeiramente superior à exatidão. Isso sugere que o modelo está a lidar bem com instâncias positivas, mas pode estar a errar mais frequentemente nas instâncias negativas.

Desempenho do modelo no conjunto de treino (alpha=1):
- Exatidão: 0.7584541062801933
- Precisão: 0.7498026585835622

Desempenho do modelo no conjunto de teste (alpha=1):
- Exatidão: 0.8142857142857143
- Precisão: 0.8292124542124543

A introdução de suavização (alpha=1) resulta em resultados similares aos obtidos com alpha=0.
O modelo ainda apresenta desempenho razoável, mas pode não ser robusto o suficiente para generalizar bem para dados não vistos.

Desempenho do modelo no conjunto de treino (alpha=3):
- Exatidão: 0.7342995169082126
- Precisão: 0.720372725871107

Desempenho do modelo no conjunto de teste (alpha=3):
- Exatidão: 0.7857142857142857
- Precisão: 0.7910052910052909

Aumentar o valor de alpha para 3 reduz ligeiramente o desempenho.
Pode indicar que uma suavização mais forte está a comprometer a capacidade do modelo de capturar padrões nos dados.

Desempenho do modelo no conjunto de treino (alpha=5):
- Exatidão: 0.7536231884057971
- Precisão: 0.7425152284889887

Desempenho do modelo no conjunto de teste (alpha=5):
- Exatidão: 0.7857142857142857
- Precisão: 0.7806122448979591

O aumento adicional de alpha para 5 não melhora o desempenho do modelo.
Pode ser que a suavização excessiva esteja a prejudicar a capacidade do modelo de discriminar entre as classes.

Concluindo, o modelo Naïve Bayes mostra consistência razoável nos resultados, mas parece sensível à escolha do parâmetro alpha.