



PRIMEIRA LISTA DE EXERCÍCIOS

Aprendizagem de Máquina (ES456/2023-1)

Aluno

Henrique Pedro da Silva

Professor

Dr. Daniel de Filgueiras Gomes

Repositório

https://github.com/Shapis/ufpe_ee

Sumário

1 Introdução	3
1.1 Conjunto de dados	3
2 Classificador Bayesiano	4
2.1 Passo a passo	4
2.1.1 Método fit	4
2.1.2 Método predict	5
2.1.3 Método _calculate_posterior	5
2.1.4 Método _calculate_likelihood	6
2.2 Resultados	6
2.2.1 Tabela de métricas	6
2.2.2 Matriz Confusão:	7
3 Classificador Método dos Mínimos Quadrados	8
3.1 Passo a passo	8
3.1.1 Método fit	8
3.1.2 Método predict	8
3.2 Resultados	9
3.2.1 Tabela de métricas	9
3.2.2 Matriz Confusão:	9
4 Conclusão	10

Tabela de Figuras

Figura 1: Fórmula da distribuição normal de Bayes.	6
---	---

Capítulo 1

Introdução

Este relatório tem como objetivo analisar um conjunto de dados utilizando dois métodos clássicos de classificação: o Classificador Bayesiano com distribuição normal e o Método dos Mínimos Quadrados com matriz pseudo-inversa.

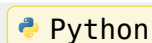
O Classificador Bayesiano assume que as variáveis seguem uma distribuição normal, calculando probabilidades para tomar decisões de classificação de forma probabilística. Já o Método dos Mínimos Quadrados minimiza os erros quadráticos entre os valores reais e preditos, sendo amplamente utilizado em problemas de regressão linear.

Para avaliar o desempenho de ambos os métodos, o conjunto de dados será dividido aleatoriamente em 80% para treinamento e 20% para teste. Assim, será possível comparar a eficácia e as características de cada abordagem na tarefa de classificação.

1.1 Conjunto de dados

Utilizaremos um conjunto de dados de 2000 elementos, e o separamos em 80% para treinamento e 20% para teste.

```
1  file_path = "dataset_ml20241212.csv"
2
3  data = pd.read_csv(file_path)
4
5  x = data.iloc[:, :-1].values
6  y = data.iloc[:, -1].values
7
8  x_train, x_test, y_train, y_test = train_test_split(
9      x, y, test_size=0.2, random_state=42
10 )
```



Capítulo 2


Classificador Bayesiano

Neste capítulo, vamos criar um classificador bayesiano e aplicá-lo ao nosso conjunto de dados.

2.1 Passo a passo

Cria-se uma classe `class NaiveBayesClassifier` que contém os métodos `fit` e `predict` para treinar e prever os dados, respectivamente.


```
1 classificador_bayes = nbc.NaiveBayesClassifier()
2 classificador_bayes.fit(x_train, y_train)
3 y_prediction_bayes = classificador_bayes.predict(x_test)
```

 Python

2.1.1 Método fit

O método `fit` calcula a média e o desvio padrão de cada classe e de cada atributo do conjunto de treinamento.

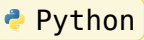
```
1 def fit(self, x, y):
2     self.classes = np.unique(y)
3     for cls in self.classes:
4         x_cls = x[y == cls]
5         self.mean[cls] = np.mean(x_cls, axis=0)
6         self.var[cls] = np.var(x_cls, axis=0)
7         self.priors[cls] = (
8             x_cls.shape[0] / x.shape[0]
9         )
```

 Python

2.1.2 Método predict

O método predict calcula a probabilidade de cada classe para cada instância do conjunto de teste e retorna a classe com maior probabilidade.

```
1 def predict(self, X):  
2     y_pred = [  
3         max(  
4             self._calculate_posterior(x),  
5             key=self._calculate_posterior(x).get  
6         )  
7         for x in X  
8     ]  
9     return np.array(y_pred)
```

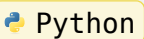


Mas para calcular a probabilidade de cada classe, é necessário calcular a probabilidade a posteriori de cada classe para cada instância do conjunto de teste.

2.1.3 Método _calculate_posterior

O método _calculate_posterior calcula a probabilidade a posteriori de cada classe para cada instância do conjunto de teste.

```
1 def _calculate_posterior(self, x):  
2     posteriors = {} # Dicionário para armazenar os posteriors  
3     # das classes  
4     for cls in self.classes:  
5         prior = np.log(self.priors[cls]) # Calcula o log da  
6         # probabilidade a priori  
7         likelihood = np.sum(  
8             np.log(  
9                 self._calculate_likelihood(self.mean[cls],  
10                 self.var[cls], x)  
11             ) # Calcula o log da verossimilhança  
12         )  
13         posteriors[cls] = (  
14             prior + likelihood  
15         ) # Calcula o posterior somando o log da prior e da  
16         # verossimilhança  
17     return posteriors # Retorna o dicionário com os posteriors  
18     # para cada classe
```



Para calcular a verossimilhança, é necessário calcular a probabilidade de cada atributo para cada classe.

2.1.4 Método `_calculate_likelihood`

O método `_calculate_likelihood` calcula a probabilidade de cada atributo para cada classe.

Para isso utiliza-se a fórmula da distribuição normal:

$$p(x|C_i) = \frac{1}{\sqrt{(2\pi\sigma)}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma}\right)^2\right]$$

Figura 1: Fórmula da distribuição normal de Bayes.

Para obter a verossimilhança, é necessário calcular o coeficiente e o expoente da distribuição normal abaixo.

```
1 def _calculate_likelihood(self, mean, var, x):  
2     eps = 1e-6 # Constante pequena para evitar problemas  
3     # numéricos por divisao por zero.  
4     coeff = 1.0 / np.sqrt(  
5         2.0 * np.pi * var + eps  
6     ) # Cálculo do coeficiente da distribuição normal  
7     exponent = np.exp(  
8         -((x - mean) ** 2) / (2 * var + eps)  
9     ) # Cálculo do expoente da distribuição normal  
10    return coeff * exponent # Retorna a verossimilhança
```

2.2 Resultados

Os resultados obtidos das métricas de acurácia e MSE para o classificador bayesiano são apresentados na tabela abaixo.

2.2.1 Tabela de métricas

Métrica	Valor
Acurácia	0.9925

Métrica	Valor
MSE	0.01
Desvio Padrão	0.09

2.2.2 Matriz Confusão:

$$\begin{pmatrix} 204 & 3 \\ 0 & 193 \end{pmatrix}$$

Capítulo 3


Classificador Método dos Mínimos Quadrados

Neste capítulo, vamos criar um classificador bayesiano e aplicá-lo ao nosso conjunto de dados.

3.1 Passo a passo

Cria-se uma classe `class LeastSquares` que contém os métodos `fit` e `predict`.


```
1 classificador_least_squares = ls.LeastSquares()  
2 classificador_least_squares.fit(x_train, y_train)  
3 y_prediction_ls = classificador_least_squares.predict(x_test)
```

 Python

3.1.1 Método fit

O método `fit` calcula os coeficientes do modelo de regressão linear utilizando o método dos mínimos quadrados por matriz pseudo inversa.


```
1 def fit(self, X, y):  
2     X_b = np.c_[np.ones((X.shape[0], 1)), X]  
3     self.theta = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

 Python

3.1.2 Método predict

O método `predict` realiza a predição do modelo de regressão linear e retorna o dicionário de predições.

```
1 def predict(self, X):  
2     X_b = np.c_[np.ones((X.shape[0], 1)), X]  
3     predictions = X_b.dot(
```

 Python


```

4         self.theta
5     )
6     return np.where(predictions <= 1.5, 1, 2)

```

3.2 Resultados

Os resultados obtidos das métricas de acurácia e MSE para o classificador do método dos mínimos quadrados são apresentados na tabela abaixo.

3.2.1 Tabela de métricas

Métrica	Valor
Acurácia	0.9925
MSE	0.01
Desvio Padrão	0.09

3.2.2 Matriz Confusão:

$$\begin{pmatrix} 204 & 3 \\ 0 & 193 \end{pmatrix}$$

Capítulo 4

Conclusão

Observou-se que os resultados obtidos com ambos os métodos foram idênticos e bastante satisfatórios, apresentando uma acurácia de 0,9925 e um erro médio quadrático (MSE) de 0,01. Esses números indicam que ambas as abordagens são altamente eficientes para a tarefa de classificação deste conjunto de dados.

Além disso, foi notado que o Método dos Mínimos Quadrados se destaca por sua simplicidade e rapidez de implementação, tornando-o uma opção prática para problemas similares. Por outro lado, o Classificador Bayesiano, embora mais complexo e demorado, pode ser preferido em situações onde a interpretação probabilística das classes seja um fator relevante.