Aprendizagem Automática
Trabalho Prático 1

```python
class NaivebayesUevora:

    # Construtor que recebe o valor de alpha e o nome do ficheiro
    def __init__(self, a, dataset):
        self.aplha = a                          # valor de alpha
        self.data = dataset                     # nome do ficheiro

    # Método que faz a leitura dos dados do ficheiro
    def read(self):
        self.table = pd.read_csv(self.data)
        #print(self.table)

    # Método que faz a separação
    def split_features_target(self):
        f = self.table.drop([self.table.columns[-1]], axis = 1)
        t = self.table[self.table.columns[-1]]
        return f, t

    # Método que calcula probabilidade com o estimador utilizado
    def calculate_prob(self, cases_number, total, num_values):
        p = (cases_number + self.aplha) / (total + (self.aplha*num_values))
        return p

    # Método que gera um classificador a partir de um conjunto de treino com etiquetas
    def fit(self, features, target):
        self.X_train = features                                 # table (possui só as caracteristicas de todos os exemplos/casos)
        self.y_train = target                                   # table (possui apenas as etiquetas "yes" / "no")
        self.examples = len(self.X_train)                       # Número de exemplos/casos (rows)
        self.features = list(features.columns)                  # Lista que contêm o nome das colunas
        self.num_values_feat = {}                               # Dicionário que guarda o número de diferentes valores possiveis de cada caracteristica/coluna
        self.num_values_targ = len(np.unique(self.y_train))     # Dicionário que guarda o número de diferentes valores possiveis na etiqueta
        self.count_yes = 0                                      # Número de yes's
        self.count_no = 0                                       # Número de no's

        for feat in self.features:
            n = len(np.unique(self.X_train[feat]))
            self.num_values_feat.update({feat : n})

        for e in range(self.examples):
            if(self.y_train.loc[e] == "yes"):
                self.count_yes+=1
            else:
                self.count_no+=1

        self.occurrences_feat_values_yes()
        self.occurrences_feat_values_no()

        p_yes = self.calculate_prob(self.count_yes,self.examples,self.num_values_targ)         # P(yes)
        p_no = self.calculate_prob(self.count_no,self.examples,self.num_values_targ)           # P(no)

        self.probability_yes(p_yes)
        self.probability_no(p_no)
```


Trabalho realizado por:
António Nanita nº 48407
Eugénio Musteata nº 45824
Rodrigo Alves nº48681

Docente:
Professor Luís Rato

1. Descrição do Trabalho

Neste trabalho pretende-se desenvolver uma classe que permita a utilização do algoritmo ***Naive Bayes*** com dados do tipo nominal, com um estimador suavizado, e avaliação do classificador através da exatidão e precisão. A classe (**NaiveBayesUevora**) implementada possui um construtor que define qual o valor de *alpha*(a) a ser utilizado como também define onde vão ser descarregados o conjunto de treino(dataset). A classe é definida pelos seguintes métodos:

- *read()*
- *split_features_target()*
- *calculate_prob(cases_number,total,num_values)*
- *fit(features,target)*
- *probability_yes(py)*
- *probability_no(pn)*
- *occurrences_feat_values_yes()*
- *occurrences_feat_values_no()*
- *predict(query)*
- *accuracy_score(X,y)*
- *precision_score(X,y)*

2. *"Main"*

No ficheiro onde foi implementada a classe **NaiveBayesUevora** foi criada uma "*main*" para testar a classe. Na "*main*" fazemos a inicialização da classe com o valor de *alpha* que pretendemos utilizar como também o conjunto de treino. De seguida fazemos a leitura do ficheiro que possui o conjunto de treino indicado, após esta operação fazemos a divisão do conjunto de treino em duas partes, "*X_train*" e "*y_train*". Após o processo de separação de dados, calculamos todas as probabilidades para aquele conjunto de treino, isto é, com os valores possíveis presentes no conjunto de treino.
Posteriormente é necessário indicar qual conjunto de teste, caso este possuir só um teste, ou seja, uma "*query*", por decisão do grupo seria apresentado apenas o valor que é calculado no método *predict*, mas se houver mais do que uma "*query*" seria apresentado a exatidão e a precisão.

3. Métodos

read() **-** Este método serve para fazer a leitura do ficheiro que possui o conjunto de treino, e converte o conteúdo do mesmo para uma "*table*".

split_features_target() - Este método serve para fazer a separação da "*table*" em duas partes e retornar as mesmas, uma delas é "*X_train*" que será uma "*table*" que possui todas as colunas expeto a última coluna da "*table*" inicial, "*y_train*" que será representada pela última coluna da "*table*" inicial.

calculate_prob(cases_number,total,num_values) - Este método serve para calcular a probabilidade com o estimador utilizado.

fit(features,target) - Este método serve para calcular todas as probabilidades para cada um dos valores possíveis de cada atributo no conjunto de treino. O grupo decidiu que para guardar estas probabilidades todas calculadas, as mesmas são guardadas num dicionário que define para cada valor possível qual a sua probabilidade no conjunto de treino, apresentando o seguinte esquema no dicionário "{valor1 : probabilidade_valor1}".

probability_yes(py) - Este método serve como método auxiliar ao método ***fit***, pois neste método são calculadas todas as probabilidades do tipo *P(_|yes)* .

probability_no(pn) - Este método serve como método auxiliar ao método ***fit***, pois neste método são calculadas todas as probabilidades do tipo *P(_|no)* .

occurrences_feat_values_yes() - Este método serve para fazer a contagem do número de casos existentes de um dado valor possível de um atributo em *"yes"*.

occurrences_feat_values_no() - Este método serve para fazer a contagem do número de casos existentes de um dado valor possível de um atributo em *"no"*.

predict(query) - Este método serve para retornar um *array*, com um tamanho igual ao número de testes, no qual estão todas as previsões que foram calculadas para cada um dos testes definidos no conjunto de teste.

accuracy_score(X,y) - Este método serve calcular a exatidão dum classificador para um dado conjunto de teste.

precision_score(X,y) - Este método serve calcular a precisão dum classificador para um dado conjunto de teste.

4. Decisões tomadas/Problemas

A primeira decisão tomada pelo grupo foi o facto de todos os dado\conteúdos de um determinado ficheiro foram guardados numa tabela, pois achamos que seria mas fácil manipular esta estrutura de dados.

De seguida, utilizamos dicionários para guardar as probabilidades para cada valor possível de cada característica, porque esta estrutura de dados permite atribuir para uma certa chave(valor possível) uma probabilidade, assim é mais fácil encontrar uma certa probabilidade a partir da chave que lhe foi atribuída.

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 numero 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 calculo da previsão só serão utilizados as probabilidades de valores que constam no conjunto de treino.

5. Análise de desempenho

Aplicando para o "*breast-cancer-train.csv*":

- Exatidão e Precisão para "*alpha*" igual 0:

Valor de exatidão = 0,62;

Valor de precisão=0.47;

- Exatidão e Precisão para "*alpha*" igual 1:

Valor de exatidão = 0.67;

Valor de precisão=0.57;

- Exatidão e Precisão para "*alpha*" igual 5:

Valor de exatidão = 0.67;

Valor de precisão=0.57;

6. Conclusão

Concluindo, neste trabalho, foram desenvolvidas mais competências na linguagem Python, como também no conhecimento deste algoritmo de forma mais aprofundada.
Ao longo deste trabalho foram encontradas algumas dificuldades, mas com alguma pesquisa foram resolvidas facilmente.