Relatório da Terceira Entrega – Preparação dos Dados

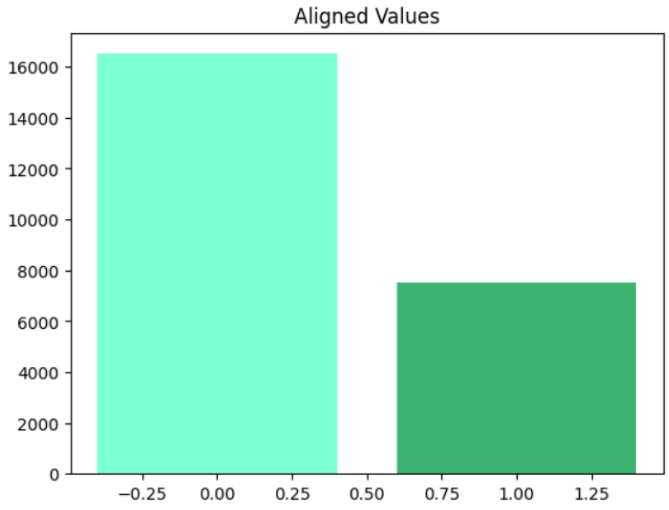
Projeto final da disciplina Soluções em Mineração de Dados

Aluno: Andrey Moutelik (arma2)

Relembrando de entregas anteriores, o dataset 'Swarm Behavior Data' é na realidade três datasets que classificam coisas diferentes: 'Aligned', 'Flocking' e 'Grouped'. Além disso 'Aligned' e 'Grouped', quando retirada a coluna de classe, tem dados identicos. Já 'Flocking' é suficientemente distinto para não podermos considera-lo no mesmo problema. É relevante comentar também que os três datasets tem 2400 features cada, e 24016 entradas. O maior desafio portanto é lidar com esse número enorme de features.

No relatório anterior explorei a ideia de fazer um classificador que conseguisse detarminar a classe dentre 4 classes distintas (uma permutação de Aligned com Grouped), mas o desbalanceamento de classes resultante foi suficiente para abandonar a ideia.

Nesta etapa do projeto, primeiramente decidi trabalhar apenas com um dos três datasets. O processo de preparação é identico para todos os três, exceto pelo fato do dataset 'Flocking' não precisar de balanceamento, pois está perfeitamente 50:50, e portanto as etapas seguintes são aplicadas apenas no dataset 'Aligned'. Este dataset é também o mais desbalanceado, e portanto a sua etapa de balanceamento será a mais relevante.




```
0    16511
1     7505
Name: Class, dtype: int64
```

Separação dos dados em subsets de treino e teste

Eu já começo o processo de preparação separando os dados em treino e teste.

```
[86] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Estudos empíricos sugerem que um split de 70:30 ou 80:20 são ideais para melhores resultados. Considerando que utilizarei Undersampling na etapa seguinte e o subaset de treinamento será reduzido, opto aqui por fazer o subset de teste 20% do total.

```
Shape of X_train: (19212, 2400)
Shape of X_test: (4804, 2400)
Shape of y_train: (19212,)
Shape of y_test: (4804,)
```

como referência, como ficou o shape dos datasets.

Boas práticas sugerem a separação do dataset em subsets de treino e teste antes da etapa de balanceamento. A razão para isso é que essas técnicas, como oversampling e undersampling, são usadas para ajustar a distribuição das classes no conjunto de dados para resolver o desequilíbrio de classes. Se aplicarmos essas técnicas a todo o conjunto de dados (incluindo o conjunto de teste), corremos o risco de introduzir viés no conjunto de teste, o que pode levar a estimativas de desempenho excessivamente otimistas.

Ao usar técnicas de balanceamento apenas no conjunto de treinamento, garantimos que nosso modelo aprenda a generalizar para dados balanceados e evite o ajuste excessivo aos dados de treinamento desbalanceados. Em seguida, avaliamos o desempenho do nosso modelo no conjunto de teste independente, que é representativo da verdadeira distribuição dos dados no mundo real.

Balanceamento

Nesta etapa, farei o balanceamento do subset de treinamento, utilizando primeiramente Undersampling, reduzindo o número de elementos da classe mais representada, e em seguida Oversampling, aumentando o número da classe menos representada. Acredito que para o meu problema essa seja a melhor estratégia, a fim de não reduzir drásticamente o número de dados, mas também não gerar muitos dados 'artificiais'.

```
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler

samplingNum = 0.7

rus = RandomUnderSampler(sampling_strategy= samplingNum, random_state=42)
X_train_resampled, y_train_resampled = rus.fit_resample(X_train, y_train)
```

```
# Apply oversampling to the resampled training set
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_resampled, y_train_resampled)
```

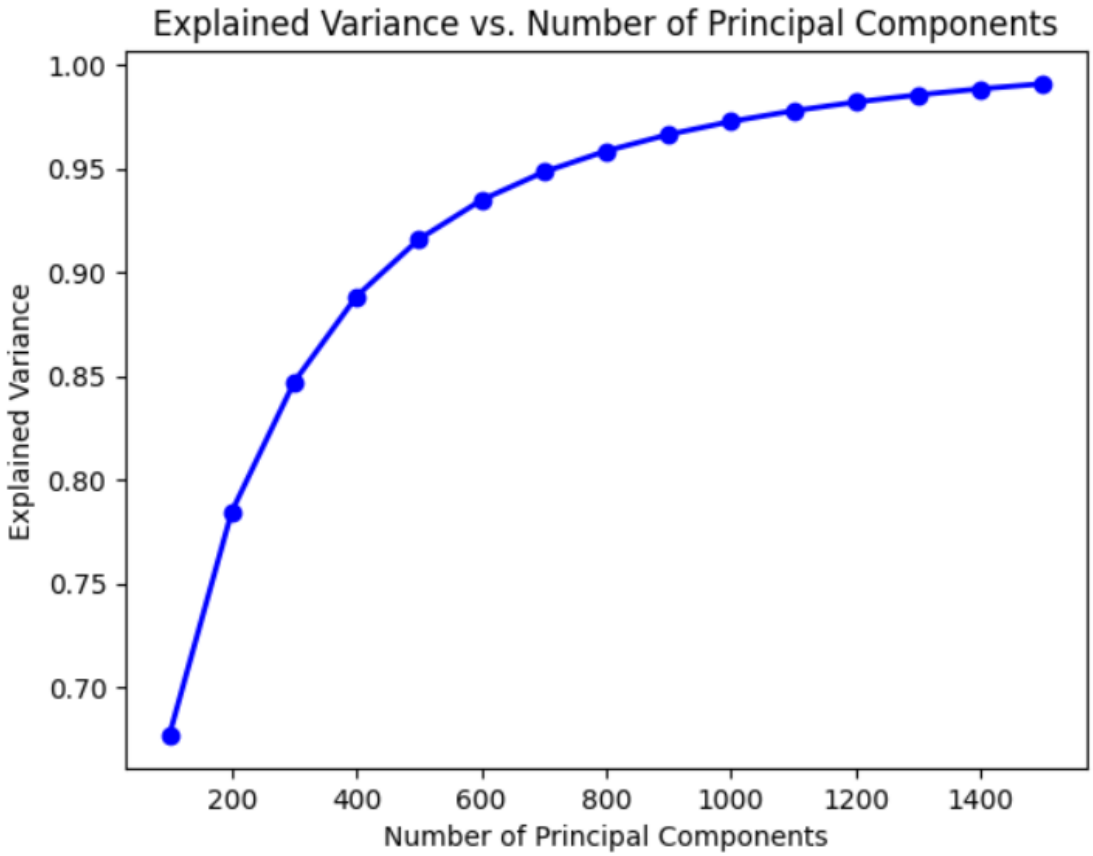
O parâmetro sampling_strategy foi escolhido pelo meu viés do que parece ser mais adequado, mas pode ser trocado depois se resultar em um desempenho melhor dos modelos.

```
Number of samples in each class after undersampling:
0    8615
1    6031
Name: Class, dtype: int64
Number of samples in each class after oversampling:
0    8615
1    8615
Name: Class, dtype: int64
```

Principal Component Analysis (PCA)

Por fim, aplico PCA afim de reduzir drasticamente o número de features do dataset. Isso é necessário para reduzir Overfitting do modelo, e para reduzir o tempo de processamento depois.

Normalizo os dados, e em seguida faço um gráfico de 'Explained Variance' x 'Number of Principal Components'. Essa etapa é a que mais leva tempo (cerca de 6 minutos no colab), portanto o número de componentes no gráfico varia de 100 a 1500, em intervalos de 100.



De acordo com o seguinte artigo sobre PCA no medium, eu devo almejar por uma métrica de 0.8, ou 80% de Explained Variance, a fim de reduzir overfitting.

https://towardsdatascience.com/dealing-with-highly-dimensional-data-using-principal-component-analysis-pca-fea1ca817fe6

So how can you tell how much information is retained in your PCA?

We use *Explained Variance Ratio* as a metric to evaluate the usefulness of your principal components and to choose how many components to use in your model. The explained variance ratio is the percentage of variance that is attributed by each of the selected components. Ideally, you would choose the number of components to include in your model by adding the explained variance ratio of each component until you reach a total of around 0.8 or 80% to avoid overfitting.

No meu caso, opto por usar 300 componentes, com uma Explained Variance de ~85%, mas logicamente esse parâmetro pode ser alterado no futuro. 300 componentes representam 12,5% do dataset original, tratando-se de uma redução drástica em sua dimensionalidade.

```
np.sum(pca.explained_variance_ratio_)
```

```
0.847151158598568
```

Com os componentes escolhidos, aplico PCA nos subsets de treino e teste

```
Shape of X_train_pca: (17230, 300)
Shape of X_test_pca: (4804, 300)
Shape of y_train_resampled: (17230,)
Shape of y_test: (4804,)
```

E por fim normalizo os dados e, a fim de facilitar a minha vida depois retorno seus nomes para o padrão:

```
X_train = scaler.fit_transform(X_train_pca)
X_test = scaler.fit_transform(X_test_pca)
y_train = y_train_resampled
```

-Andrey Moutelik