Faculdade XPe

Capítulo 03

MÉTODOS ENSEMBLE

Silas Liu

2024

Vantagens e aplicações de métodos ensemble

Os Métodos Ensemble são uma abordagem poderosa em Machine Learning, com diversas vantagens. A ideia principal por trás desses métodos é combinar múltiplos modelos para obter um desempenho melhor do que apenas um modelo individual. Essa combinação de modelos ajuda a capturar diferentes padrões nos dados, reduzir o viés e a variância e, consequentemente, melhorar a precisão e a robustez das previsões. Vamos detalhar as principais vantagens dos métodos ensemble:

- **Maior precisão:** uma das principais vantagens dos métodos ensemble é sua capacidade de melhorar a acurácia do resultado. Ao combinar diferentes modelos, o ensemble consegue capturar nuances que um único modelo poderia perder. Isso é especialmente útil em cenários onde os dados são complexos e apresentam muitas variáveis ou interações não lineares.
- **Redução de overfitting:** alguns algoritmos de ML, como árvores de decisão, podem sofrer de overfitting quando são muito complexos e perdem a capacidade de generalizar suas previsões para novos dados. Métodos ensemble ajudam a reduzir esse risco, uma vez que combinam várias versões de modelos, suavizando resultados com overfitting.
- **Robustez:** os métodos ensemble são menos suscetíveis a erros causados por ruído nos dados. Isso ocorre porque erros de modelos individuais podem se cancelar mutuamente quando combinados.
- **Flexibilidade:** os métodos ensemble podem ser aplicados a uma ampla gama de modelos base. Eles funcionam bem tanto com algoritmos de baixa complexidade, como regressões logísticas, quanto modelos mais complexos, como redes neurais.

Em resumo, as vantagens dos métodos ensemble tornam essa abordagem extremamente valiosa. Obviamente, métodos ensemble trazem um custo por trás, que refletem em mais processamento e tempo para treinar os modelos.

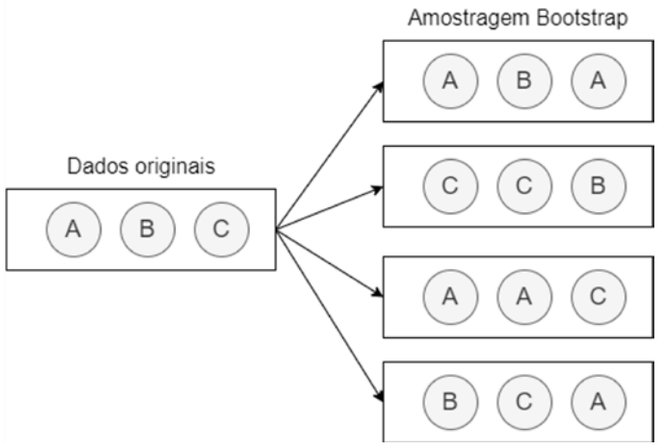
Bagging

Bagging é um dos métodos ensemble mais populares e representa a abreviação do nome Bootstrap Aggregating. Nesse método, a ideia é em cada modelo base treinar usando subsets diferentes dos dados. Agregando todos os modelos, conseguimos reduzir a variância e aumentar a estabilidade das previsões. O modelo é mais simples por não envolver hiperparâmetros. A seguir seguem os detalhes de cada etapa do método Bagging:

Amostragem bootstrap:

Para cada modelo, os dados originais são amostrados de forma aleatória, com reposição. Isso quer dizer que dados podem se repetir e outros dados podem não ser usados.

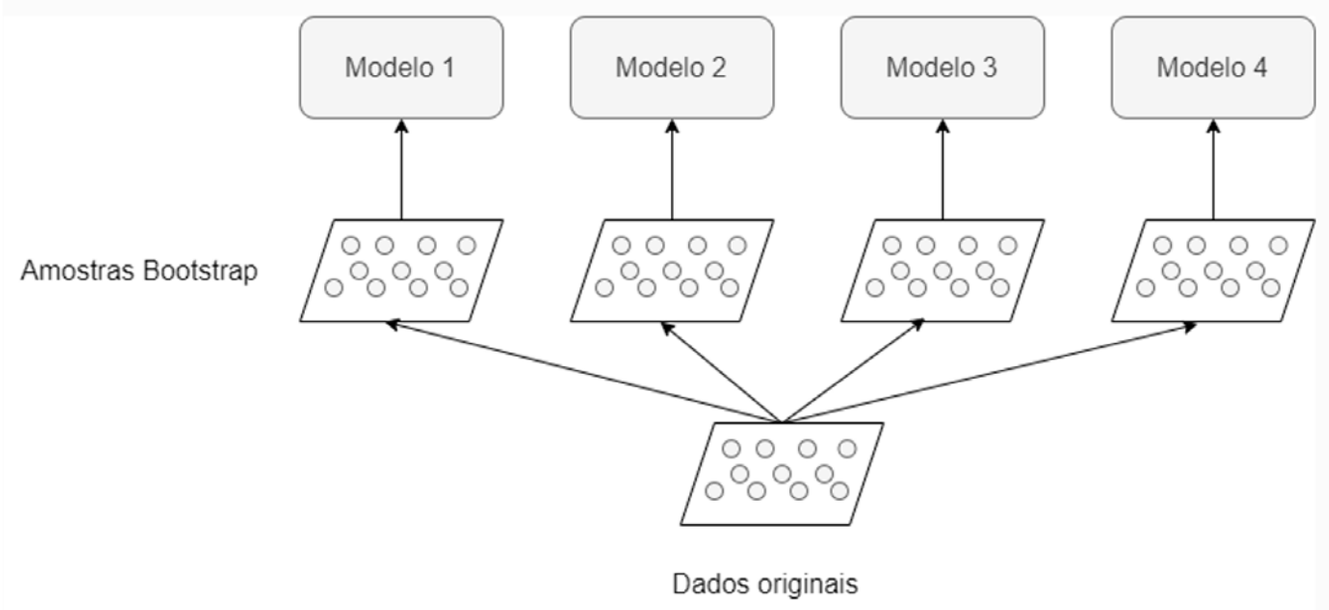
Figura 1 – Amostragem bootstrap do método bagging.



Treinamento:

Para cada modelo, os dados originais são amostrados de forma aleatória, com reposição. Isso quer dizer que dados podem se repetir e outros dados podem não ser usados.

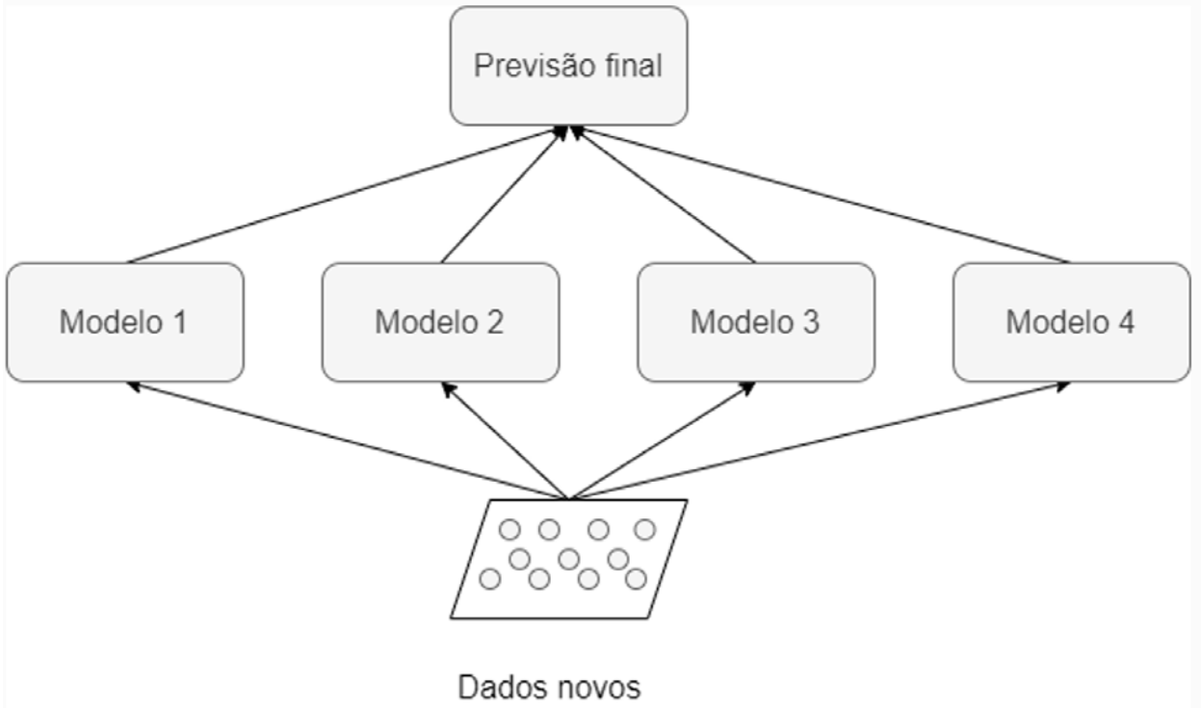
Figura 2 – Treinamento do método Bagging.



Previsão:

Na etapa de previsão do método Bagging não há amostragem dos dados. Para problemas de classificação, o resultado é escolhido por votação majoritária (a classe mais comum entre as previsões). Para problemas de regressão, o resultado é a média das previsões.

Figura 3 – Previsão do método Bagging.



O uso do método Bagging via código, pela biblioteca scikit-learn é bem simples. Ele funciona como os outros modelos, tem as funções fit para treinar e predict para prever. No exemplo a seguir mostramos o uso para um

Figura 4 – Exemplo de código no scikit-learn para o método bagging.

```
# Importação das bibliotecas
from sklearn.ensemble import BaggingClassifier, BaggingRegressor
from sklearn.tree import DecisionTreeeClassifier
from sklearn.model_selection import train_test_split
```

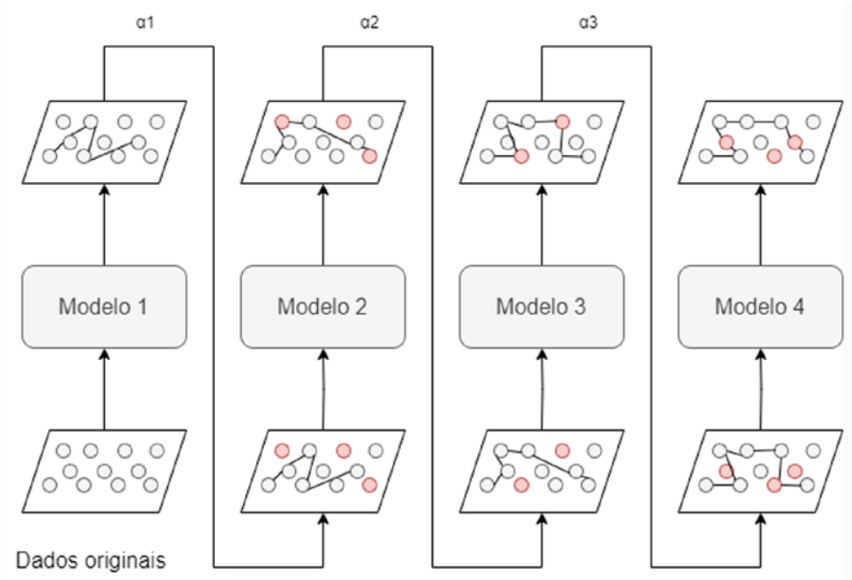
```
# Separação em sets de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Declara o modelo base
base_model = DecisionTreeClassifier(max_depth=4,
                                    min_samples_leaf=0.16)
# Instancia o ensemble Bagging
ensemble = BaggingClassifier(base_estimator=base_model,
                             n_estimators=100)
# Treina o ensemble Bagging
ensemble.fit(X_train, y_train)
# Previsão usando o Bagging
y_pred = ensemble.predict(X_test)
```

AdaBoost

AdaBoost é um método ensemble baseado na técnica boosting, muito popular também e representa a abreviação do nome Adaptive Boosting. O conceito por trás dessa técnica é transformar modelos “fracos” em modelos “fortes”. A metodologia consiste em treinar vários modelos em série, com pesos diferentes e que se adaptam (de onde vem o termo adaptativo). A cada passagem atribui-se pesos diferentes para cada ponto dos dados, dando mais peso àqueles que tiveram erro maior na predição do modelo anterior. Isso representa que a cada modelo, ele é treinado com mais ênfase nos erros do anterior. Além disso os modelos que se sobressaem melhor recebem um peso maior na contribuição do resultado.

É importante esclarecer que esses pesos adaptativos são aplicados apenas na fase de treinamento. Durante a predição, os dados passam por todos os modelos, mas cada modelo tem sua parcela de contribuição, determinada pela performance durante o treinamento.

Figura 5 – Treinamento do método AdaBoost.



A seguir seguem os códigos para um exemplo de aplicação com uma árvore de decisão.

Figura 6 – Exemplo de código no scikit-learn para o método AdaBoost.

```
# Importação das bibliotecas
from sklearn.ensemble import AdaBoostClassifier, AdaBoostRegressor
from sklearn.tree import DecisionTreeeClassifier
from sklearn.model_selection import train_test_split
```

```
# Separação em sets de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Declara o modelo base
base_model = DecisionTreeClassifier(max_depth=1)

# Instancia o ensemble AdaBoost
ensemble = AdaBoostClassifier(base_estimator=base_model,
                              n_estimators=100)
# Treina o ensemble AdaBoost
ensemble.fit(X_train, y_train)
# Previsão usando o AdaBoost
y_pred = ensemble.predict(X_test)
```

Gradient Boosting

Gradient Boosting é outra técnica ensemble baseada no Boosting. Assim como o modelo anterior, esse modelo também concatena os modelos em série, sendo um treinado após o outro. A diferença é que nesse método, ao invés do modelo tentar prever o y, ele usa o erro residual de previsão do modelo anterior (r=y-y^) como variável de interesse, nesse caso ele tenta minimizar os erros residuais para 0. A seguir seguem o esquemático de treinamento, bem como exemplo de código no Python. Uma diferença dos demais é que esse método funciona apenas com árvore de decisão como modelo base, e este não precisa ser declarado no código.

Figura 7 – Treinamento do método Gradient Boosting.

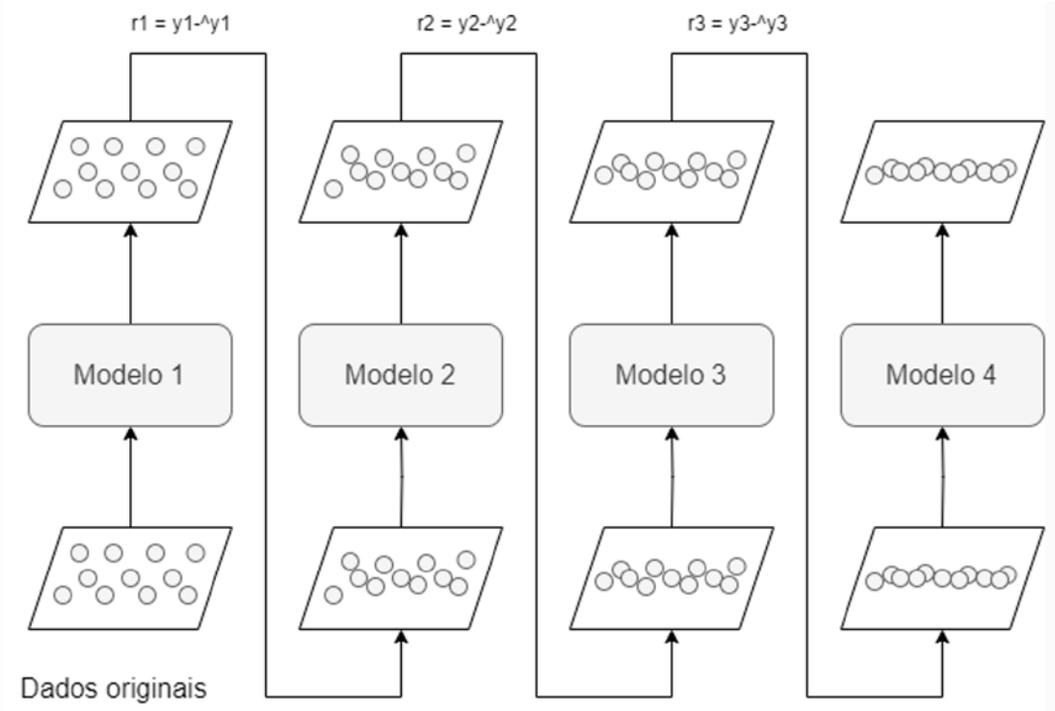


Figura 8 – Exemplo de código no scikit-learn para o método Gradient Boosting.

```
# Importação das bibliotecas
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.model_selection import train_test_split
```

```
# Separação em sets de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Instancia o ensemble Gradient Boosting
ensemble = GradientBoostingClassifier(n_estimators=300,
                                      max_depth=1)
# Treina o ensemble Gradient Boosting
ensemble.fit(X_train, y_train)
# Previsão usando o Gradient Boosting
y_pred = ensemble.predict(X_test)
```

Stochastic Gradient Boosting

O método Stochastic Gradient Boosting se baseia no mesmo conceito do método anterior, o Gradient Boosting, mas acrescenta uma camada de amostragem e aleatoriedade, o que garante modelos mais versáteis. Durante o treinamento, duas metodologias são aplicadas simultaneamente. Cada modelo é treinado em um subset amostrado aleatoriamente dos dados, sem reposição. Para a construção das árvores de decisão de cada modelo, a cada nova folha, um subset das colunas é amostrado aleatoriamente para serem usados como features. Dessa maneira, garante-se treinamento em sets bastante diferentes, dos dados originais.

Na biblioteca sklearn, utilizamos a mesma classe do Gradient Boosting, mas usamos os parâmetros subsample e max_features para controlar a parcela que será usada, dos dados e features, respectivamente.

Figura 9 – Treinamento do método Stochastic Gradient Boosting

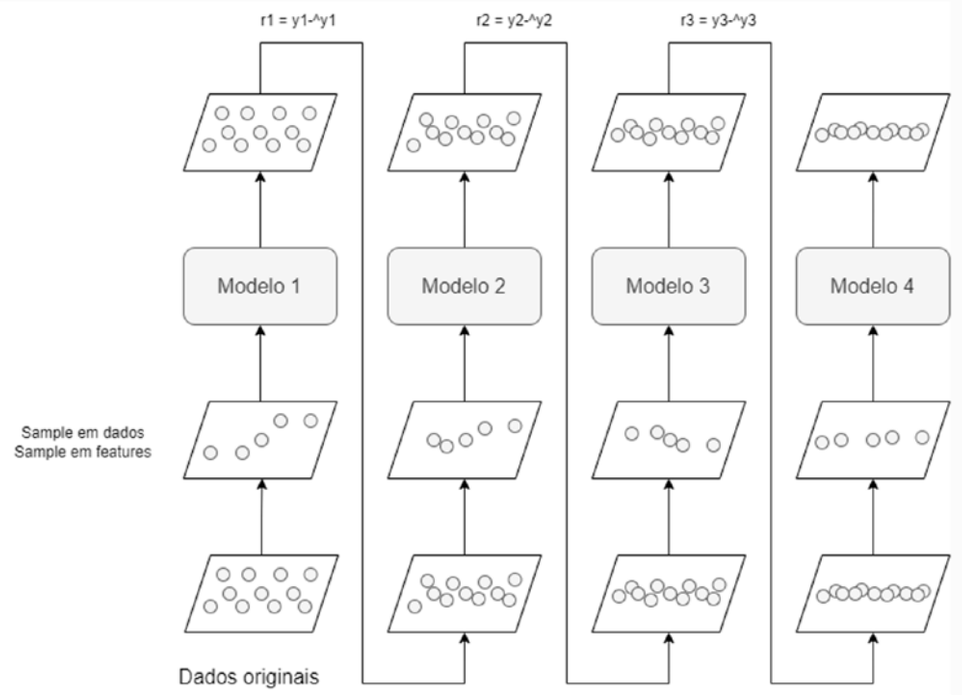


Figura 10 – Exemplo de código no scikit-learn para o método Stochastic Gradient Boosting.

```
# Separação em sets de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Instancia o ensemble Stochastic Gradient Boosting
ensemble = GradientBoostingClassifier(n_estimators=300,
                                      max_depth=1,
                                      subsamples=0.8,
                                      max_features=0.2)
# Treina o ensemble Stochastic Gradient Boosting
ensemble.fit(X_train, y_train)
# Previsão usando o Stochastic Gradient Boosting
y_pred = ensemble.predict(X_test)
```

XGBoost

XGBoost (Extreme Gradient Boosting), é uma versão otimizada do Gradient Boosting, projetada para ser extremamente rápido e eficiente. Ele utiliza o mesmo conceito básico do Gradient Boosting, onde modelos são treinados sequencialmente sobre os erros residuais dos modelos anteriores. Entretanto, esse modelo traz diversas inovações, como regularização L1 e L2 para evitar overfitting, manejo automático de dados faltantes e paralelização no treinamento dos modelos. XGBoost também implementa o efeito “shrinkage”, que controla a contribuição de cada árvore e o “early stopping”, que interrompe o treinamento quando a métrica atinge valores aceitáveis.

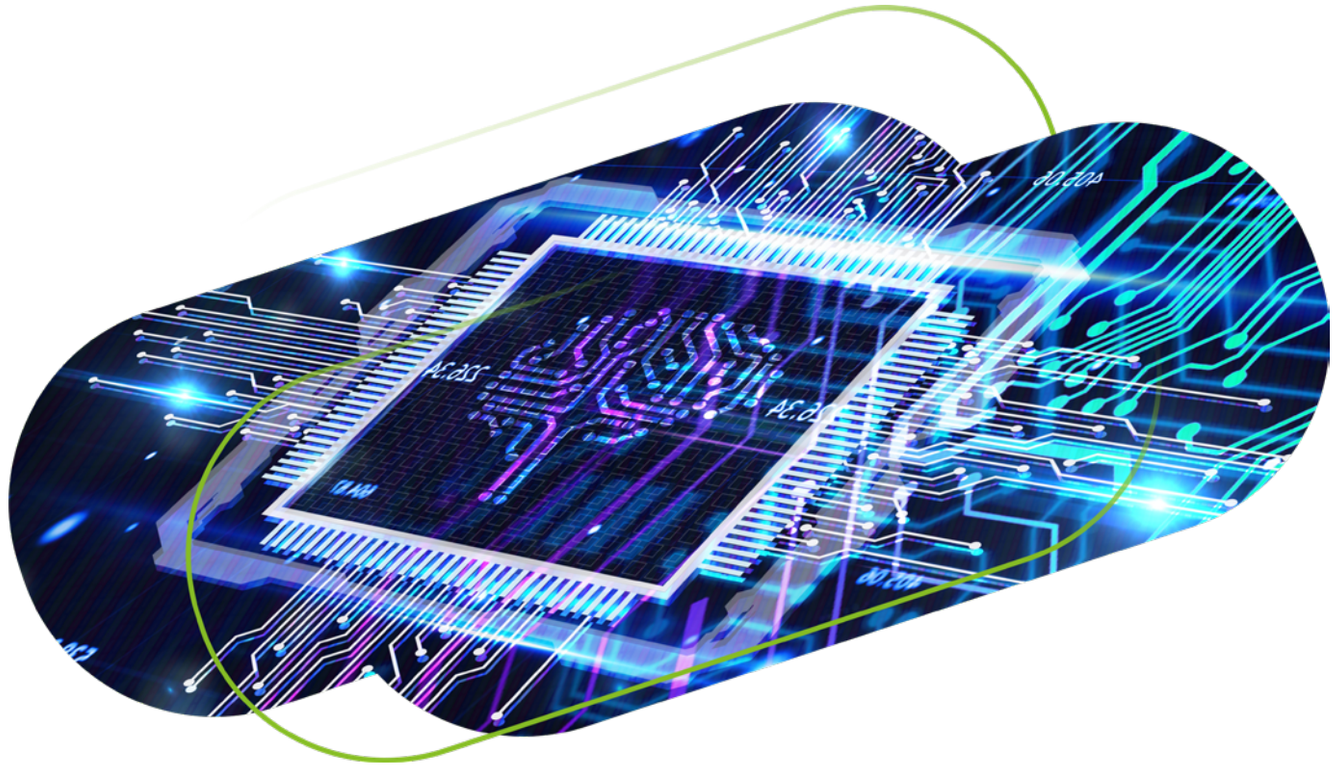
O XGBoost possui sua própria biblioteca, o xgboost, mas ele pode ser usado tanto sozinho como junto à pipelines do scikit-learn.

Figura 11 – Exemplo de código no scikit-learn para o método XGBoost.

```
# Importação das bibliotecas
import xgboost as xgb # xgb.XGBClassifier ou xgb.XGBRegressor
from sklearn.model_selection import train_test_split
```

```
# Separação em sets de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Instancia o ensemble XGBoost
ensemble = xgb.XGBClassifier()
# Treina o ensemble XGBoost
ensemble.fit(X_train, y_train)
# Previsão usando o XGBoost
y_pred = ensemble.predict(X_test)
```

Demonstramos aqui diversas técnicas diferentes de técnicas ensemble. Entretanto, cada caso é um caso e não existe um melhor modelo para todos os casos de uso. Os dados e a modelagem do problema devem ser analisados com cautela, bem como a análise da quantidade de dados, quantidade de features, distribuição de valores e a métrica utilizada.



Referências

AMERICAN National Standards Institute. The SQL Standard – ISO/IEC 9075:2016. Disponível em: <https://blog.ansi.org/2018/10/sql-standard-iso-iec-9075-2016-ansi-x3-135/>. Último acesso: 04 de abr, 2020.

CHAMBERLIN, Donald D.; BOYCE, Raymond F. SEQUEL: A Structured English Query Language. Disponível em: <http://www.joakimdalby.dk/HTM/sequel.pdf>. Último acesso: 04 de abr, 2020.

CHEN, Peter. Modelagem de Dados. São Paulo: McGraw-Hill, Makron, 1990.

CHEN, Peter. Peter Chen Home Page at Louisiana State University (LSU). Disponível em: <https://www.csc.lsu.edu/~chen>. Último acesso: 01 de mai, 2020.

CHEN, Peter. The Entity-Relationship Model - Toward a Unified View of Data. Disponível em: <https://dspace.mit.edu/bitstream/handle/1721.1/47432/entityrelationshx00chen.pdf&>. Último acesso: 01 de mar, 2020.

CODD, Edgar F. A Relational Model of Data for Large Shared Data Banks.

Disponível em: <https://github.com/dmvaldman/library/blob/master/computer%20science/Codd%20-%20A%20Relational%20Model%20of%20Data%20for%20Large%20Shared%20Data%20Banks.pdf> . Último acesso: 03 de mar, 2020.

CODD, Edgar F. The Relational Model for Database Management: Version 2. United States Of America: Addison Wesley Publishing Company, 1990.

CODD's Twelve Rules. Disponível em: <https://computing.derby.ac.uk/c/codds-twelve-rules>. Último acesso: 09 de mar, 2020.

COUGO, Paulo. Modelagem Conceitual e Projeto de Banco de Dados. 14 reimpressão. Rio de Janeiro: Elsevier, 1997.

GARTNER, INC. Big Data. Disponível em: <https://www.gartner.com/en/information-technology/glossary/big-data>. Último acesso: 02 de mai, 2020.

HILLS, Ted. NOSQL and SQL Data Modeling: Bringing Together Data, Semantics, and Software. Technics Publications, 2016.

IBM. A History and Evaluation of System R. Disponível em: <https://people.eecs.berkeley.edu/~brewer/cs262/SystemR.pdf>. Último acesso: 04 de abr, 2020.

KORTH, Henry F. Sistema de bancos de dados. 3. ed. São Paulo: McGraw Hill, 1999.

MACHADO, Felipe Nery Rodrigues; ABREU, Maurício Pereira de. Projeto de Banco de Dados: uma visão prática. 13. ed. São Paulo: Érica, 1996.

MCKINSEY Digital. Disponível em <https://www.mckinsey.com/business-

functions/mckinsey-digital/our-insights/big-data-the-next-frontier-for-innovation>. Último acesso: 02 de mai, 2021.

MICHAELIS. Dicionário da Língua Portuguesa Brasileira. Disponível em: <https://michaelis.uol.com.br/moderno-portugues>. Último acesso: 05 de abr, 2019.

MICROSOFT SQL Server 2019. Disponível em <https://www.microsoft.com/pt-br/sql-server/sql-server-2019>. Último acesso: 14 de out, 2022.

MICROSOFT SQL Documentation. Disponível em: <https://docs.microsoft.com/en-us/sql/?view=sql-server-2019>. Último acesso: 14 de out, 2022.

MONQUEIRO, Júlio Cesar Bessa. Programação Orientada a Objetos: uma

introdução. Disponível em: <https://www.hardware.com.br/artigos/programacao-orientada-objetos>. Último acesso: 05 de mar, 2019.

MONGODB. Site Oficial do Fabricante. Disponível em: <https://www.mongodb.com>. Último acesso: 17 de out, 2020.

MONTEIRO, Leandro Pinho. Dados Estruturados e Não Estruturados. Universidade da Tecnologia, 2019. Disponível em: <https://universidadedatecnologia.com.br/dados-estruturados-e-nao-estruturados>. Último acesso: 02 de mai, 2021.

NAVATHE, Shamkant B.; ELSMARI, Ramez. Sistemas de Banco de Dados. 4. ed. São Paulo: Pearson Addison Wesley, 2005.

NOSQL Database Org. Disponível em: <http://nosql-database.org/>. Último acesso: 16 de mar, 2019.

PAT RESEARCH. Top 9 Object Databases. Disponível em: <https://www.predictiveanalyticstoday.com/top-object-databases>. Último acesso: 05 de mar, 2020.

SIMPLILEARN. Introduction to NOSQL Databases Tutorial. Disponível em: <https://www.simplilearn.com/introduction-to-nosql-databases-tutorial-video>. Último acesso: 13 de mar, 2019.

STROZZI, Carlo. NoSQL: A non-SQL RDBMS. Disponível em: <http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page>. Último acesso: 11 de mar, 2019.

TAYLOR, Christiane. Structured vs. Unstructured Data. Disponível em:

<https://www.datamation.com/big-data/structured-vs-unstructured-data.html>. Último acesso: 04 de mai, 2021.

UNQL QUERY LANGUAGE UNVEILED BY COUCHBASE AND SQLITE. Disponível em: <https://www.couchbase.com/press-releases/unql-querylanguage>. Último acesso: 11 de mar, 2019.

UNSQL Org. Disponível em: <http://unql.sqlite.org/index.html/wiki?name=UnQL>. Último acesso: 11 de mar, 2019.

Faculdade XPe

xpeducacao.com.br