# Projeto Final A3: Desenvolvimento de um Agente Inteligente

Aluno: Marino Gonçalves Neto

## Objetivo

Desenvolver, individualmente, um agente inteligente capaz de resolver um problema específico utilizando técnicas de aprendizado de máquina. O agente deverá processar dados, aprender com eles e tomar decisões com base no aprendizado.

## Relatório Final

### Descrição do Problema

O objetivo deste projeto é desenvolver um Agente Inteligente com capacidade de classificar automaticamente sentimentos positivos e negativos expressos em tweets da rede social **Twitter** (atual **X**) através de aprendizado de máquina, que pode evoluir futuramente para demais textos de outras redes sociais ou afins e outros tipos de sentimentos. Este desafio é bastante significativo e complexo visto que os tweets são produções textuais escritas de maneira bastante "humanizada" considerando que por ser uma rede social, é costume o uso de gírias, emojis, termos utilizados apenas na internet ou até mesmo pouco utilizados na internet, usuários de diferentes faixas etárias, entre outros fatores.

A classificação de sentimentos pode ser considerada um problema bastante comum em processamento de linguagem natural e pode ser utilizada em diferentes contextos de aplicações como análise e estatísticas de opiniões de usuários em redes sociais, segurança e filtros de linguagem, monitoramento de marcas e companhias, automação de atendimento ao cliente, chatbots, entre outros.

### Dataset

O conjunto de dados utilizado foi o **Sentiment140**. Ele contém cerca de 1.6 milhão de tweets extraídos através da API do **Twitter**. São anotados e identificados como positivos e negativos para o campo "sentiment" com os valores:

- **0:** Sentimento Negativo;
- **4:** Sentimento Positivo.

Durante o pré-processamento, os valores 4 foram convertidos para 1 no intuito de facilitar a legibilidade através de um conjunto binário com duas classes: 0 (negativo) e 1 (positivo).

A quantidade total de registros estava levando cerca de mais de 1 hora e 30 minutos e para disponibilizar no github o arquivo .csv estava grande demais, então a quantidade utilizada para o teste foi reduzida e balanceada para cerca de 300.000 ao todo (150.000 positivos e 150.000 negativos), o que reduziu para até 1 minuto apenas com Random Forest (Com os demais modelos incluídos leva cerca de 2 minutos e meio a 3 minutos em média).

### Pré-processamento

O pré-processamento foi realizado com objetivo de garantir a qualidade dos dados e prepará-los para o treinamento do(s) modelo(s). As seguintes etapas foram realizadas:

- **Limpeza de Texto:**
  - Foram realizadas remoções de URLs, menções a usuários (exemplo: @usuário), caracteres especiais e numéricos;
  - Normalização de texto convertendo todas as palavras para minúsculas.
- **Lematização:**
  - Realizada a lematização com a biblioteca NLTK para reduzir as palavras às suas raízes. Exemplo: "Correndo" para "correr".
- **Vetorização:**
  - A transformação do texto em dados numéricos foi realizada utilizado o **TfidfVectorizer** com n-grams de 1 a 2 e um limite de 500.000 características. Isto foi bastante benéfico na captura de relações entre as palavras no texto.
- **Divisão em conjuntos de treinamento e teste:**
  - O conjunto foi dividido em 80% de treinamento e 20% de teste inicialmente e reajustado algumas vezes até o resultado: 70% de treinamento e 30% de teste.

### Resultados do Modelo Base

Como modelo base foi utilizado o Random Forest. Esta decisão foi tomada considerando sua popularidade e resultados para tarefas de classificação. Inicialmente foi utilizada uma taxa de 80% de treinamento e 20% de teste e após o treinamento os resultados obtidos com o modelo base foram:

- **Acurácia:** 73%;
- **Precisão:** 73% para ambos os sentimentos (positivo e negativo);
- **Recall:** 73% para ambos os sentimentos;
- **F1-score:** 73% para ambos os sentimentos.

Foi concluído que apesar de ser apenas a base inicial, o modelo estava realizando uma classificação relativamente aceitável e equilibrado visto que ambas as classes (0 e 1) têm a mesma acurácia.

### Resultados do Modelo Aprimorado

Em busca de melhorias, foram efetuadas diversas tentativas de aperfeiçoar o Agente Inteligente e consequentemente melhorar sua performance e acurácia de maneira geral. Durante este processo, foram implementados novos tipos de pré-processamentos como:

- Substituição de emojis por representações textuais. Ex: :smile: para representar o emoji de sorriso;
- Expansão de abreviações. Ex: btw = by the way;
- Substituição de URL;

- Substituição de menções de usuários da rede social para a palavra default USER;
- Lematização;
- Remoção de pontuações;
- Normalização de espaços;
- Remoção de StopWords.

Após estas adições, foi notado que a acurácia não melhorou, mas sim piorou de 73% para 69%. Considerando isto, não fazia muito sentido em meus pensamentos e busquei entender melhor a respeito. Foi neste momento que comecei a entender que algumas StopWords podem ser essenciais para o correto entendimento deste tipo de classificação... afinal, a frase "Não gosto de chocolate" é negativa e com a remoção de suas stop words ficaria algo como: "Gosto chocolate", que tem sentimento positivo. Cenários como este poderiam confundir ainda mais o Agente e por isto foi optado pela remoção deste pré-processamento relacionado a remoção de StopWords e com isto os resultados obtidos foram:

- **Acurácia:** 74%;
- **Precisão:** 76% para negativo e 73% para positivo;
- **Recall:** 71% para negativo e 77% para positivo;
- **F1-score:** 73% para negativo e 75% para positivo.

Apesar de aparecer uma oscilação após os ajustes de pré-processamento, muitos dos dados subiram e considerei isto um avanço momentâneo.

Como não estava satisfeito, busquei adicionar técnicas de ajuste de hiperparâmetros como Grid Search e Random Search, mas infelizmente o tempo de execução saltou de cerca de 1 minuto para mais de 30 minutos e a acurácia diminuiu novamente com ambas as técnicas. Entendi que a demora se dava com relação ao uso de memória e à grande quantidade de dados (160.000), então optei por reduzir a quantidade de dados aos poucos e diferentes quantidades foram testadas e reduzidas consecutivamente até chegarmos em torno de 5.000... em todos os testes apesar do tempo ser reduzido, a acurácia permanecia diminuindo e concluí que quanto menos dados, menos acurácia e precisão o Agente tinha.

Também foram procurados ajustes finos em parâmetros como n_estimators e max_dept, mas não obtive sucesso. Considerando isto, a ideia de uso com Grid Search ou Random Search foi abortada e outro plano foi seguido: Uso de outros modelos.

Com o novo plano de utilizar outros modelos ao invés de apenas o RandomForest, foram feitas algumas pesquisas e os seguintes modelos foram implementados:

- **KNN**:
  - **Acurácia**: 70%;
  - **Precisão**: 76% para negativo e 66% para positivo;
  - **Recall**: 58% para negativo e 81% para positivo;
  - **F1-score**: 66% para negativo e 73% para positivo.
- **KNN_Uniform**:
  - **Acurácia**: 70%;
  - **Precisão**: 76% para negativo e 66% para positivo;
  - **Recall**: 58% para negativo e 81% para positivo;
  - **F1-score**: 66% para negativo e 73% para positivo.
- **KNN_Distance**:
  - **Acurácia**: 70%;
  - **Precisão**: 76% para negativo e 66% para positivo;
  - **Recall**: 58% para negativo e 81% para positivo;
  - **F1-score**: 66% para negativo e 73% para positivo.
- **AdaBoost**:
  - **Acurácia**: 70%;
  - **Precisão**: 69% para negativo e 71% para positivo;
  - **Recall**: 73% para negativo e 66% para positivo;
  - **F1-score**: 71% para negativo e 68% para positivo.
- **DecisionTree**:
  - **Acurácia**: 65%;
  - **Precisão**: 63% para negativo e 67% para positivo;
  - **Recall**: 71% para negativo e 58% para positivo;
  - **F1-score**: 67% para negativo e 62% para positivo.
- **LinearSVC**:
  - **Acurácia**: 80%;
  - **Precisão**: 80% para negativo e 80% para positivo;
  - **Recall**: 80% para negativo e 80% para positivo;
  - **F1-score**: 80% para negativo e 80% para positivo.
- **LogisticRegression**:
  - **Acurácia**: 81%;
  - **Precisão**: 81% para negativo e 81% para positivo;
  - **Recall**: 81% para negativo e 81% para positivo;
  - **F1-score**: 81% para negativo e 81% para positivo.
- **BernoulliNB**:
  - **Acurácia**: 79%;
  - **Precisão**: 79% para negativo e 78% para positivo;
  - **Recall**: 78% para negativo e 80% para positivo;
  - **F1-score**: 79% para negativo e 79% para positivo.

Os modelos **GradientBoosting**, **LightGBM**, **XGBoost**, **CatBoost**, **ExtraTrees_Gini**, ExtraTrees_Entropy e **Bagging** também foram testados, mas devido a extrema demora de execução não foram obtidos resultados.

Os resultados dos demais modelos foram bastante interessantes e muitos se mostraram melhores performaticamente do que o inicial RandomForest. Após alguns momentos de estudos a respeito de cada modelo, a conclusão foi de que aparentemente os modelos mais simples como os lineares (LinearSVC e LogisticRegression) se mostraram superiores em desempenho do que os mais avançados. Provavelmente fatores como sensibilidade a ruídos e superajuste de dados podem ter reduzido o desempenho de alguns modelos como KNN ou baseados em Árvores. Já a respeito dos extremamente demorados, em sua grande maioria eram os mais avançados e devido a isto tem um custo computacional e performático mais caro para o ambiente, então a grande quantidade de dados realmente não me apoiou muito neste cenário (foram feitos testes com dados menores, mas o resultado realmente não se mostrou muito satisfatório com menos dados disponíveis mesmo com estes modelos).

Eu já me encontrava um pouco mais satisfeito com estes números em torno de 81% de acurácia, mas buscava algo ainda melhor e encontrei os metamodelos, mais especificamente:

- **VotingClassifier**: Consiste em treinar múltiplos modelos e usar uma votação para escolher a classe final;
- **StackingClassifier**: Usa os modelos base como uma etapa de pré-processamento e depois treina um modelo final.

Todos os modelos anteriores foram utilizados de diferentes maneiras para os testes com ambos os metamodelos, o tempo de treino passou de 1 hora para ambos e todos os outros cenários já testados foram reutilizados aqui para procurar um melhor cenário, mas todos falharam em desempenhar melhor e o resultado máximo alcançado foi em torno de 76% de eficácia. Devido ao longo tempo de espera e resultado insatisfatório, a ideia de metamodelo foi abortada e o código voltou para a etapa de modelos individuais, mantendo apenas os modelos com melhor performance (BernoulliNB, LinearSVC e LogisticRegression) e o RandomForest que era o modelo base.

Neste momento, não havia mais ideias de como melhorar a performance e após pesquisar a respeito, percebi que grande parte das pessoas que procuraram algo semelhante também se aproximavam de uma acurácia máxima entre 76% e 83% inclusive utilizando toda a base de 1.6 milhão de tweets.

Concluí que minha base que estava sendo executada em um tempo médio de até 3 minutos com quantidade de dados significativamente reduzida estava com valores bastante próximos (se não máximos) do esperado e me senti satisfeito com os resultados obtidos por este projeto inicial.

Os melhores modelos se mostraram ser os Lineares LinearSVC e LogisticRegression com respectivas 80% e 81% de acurácia.

### Limitações do Agente

- **Quantidade massiva de dados:** A grande quantidade de dados se mostrou um limitador bastante significativo, visto que muitas vezes me encontrei com problemas relacionados a memória ou tempo de execução altamente elevado;
- **Sensibilidade a gírias e expressões regionais:** Gírias inseridas manualmente limitam o modelo e novas gírias podem dificultar a interpretação do agente;
- **Idioma limitado ao inglês:** Atualmente, o agente está limitado ao idioma inglês devido à base de dados utilizada;
- **Limitado à rede social Twitter (atual X):** O pré-processamento está baseado em padrões específicos desta rede social.

### Possíveis Melhorias Futuras

- **Treinamento em dados mais diversificados:** Expandir o treinamento para tweets de diferentes idiomas, regiões e redes sociais;
- **Otimização do agente:** Reduzir o custo computacional para viabilizar o uso de modelos mais avançados;
- **Aprimoramento do pré-processamento:** Incorporar ferramentas de correção ortográfica, bancos de dados para emojis e gírias para melhor manutenibilidade;
- **Incorporação de demais sentimentos:** Expandir as classes para incluir sentimentos neutros e demais possibilidades de sentimentos;
- **Automação de hiperparâmetros:*** Implementar ferramentas para ajuste automático de hiperparâmetros.