Classificação de crimes em São Francisco

GUILHERME AUGUSTO ANÍCIO DRUMMOND DO NASCIMENTO and HENRIQUE DANTAS PIGHINI, Universidade Federal de Ouro Preto, Brasil

Este artigo destaca a participação em um desafio Kaggle focalizado na classificação de crimes em São Francisco, enfatizando a aplicação de técnicas de mineração de dados. A abordagem envolveu o uso de Support Vector Machines (SVM), Random Forest e Redes Neurais. A SVM foi selecionada pela sua capacidade de lidar com conjuntos de dados intricados, o Random Forest proporcionou robustez diante da diversidade dos dados, enquanto as Redes Neurais foram empregadas para capturar relações não-lineares.

CCS Concepts: • **Computing methodologies** → *Supervised learning*; Classification and regression trees; Support vector machines; Neural networks; • **Information systems** → **Data mining**.

Additional Key Words and Phrases: Do, Not, Use, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Guilherme Augusto Anício Drummond do Nascimento and Henrique Dantas Pighini. 2024. Classificação de crimes em São Francisco. 1, 1 (February 2024), 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUÇÃO

Este artigo aborda o desafio de classificação de crimes na cidade de São Francisco[Kan 2015], Califórnia, através da análise de um conjunto de dados disponibilizado no Kaggle. O objetivo principal é aplicar técnicas avançadas de mineração de dados para desenvolver modelos preditivos capazes de classificar diferentes tipos de crimes com base em variáveis diversas, como localização, horário e outras características relevantes.

O conjunto de dados em questão contém dados como dia da semana, descrição e endereço aproximado, entre outros, dos incidentes criminais ocorridos em São Francisco, abarcando o período de 01/01/2003 a 13/05/2015. O objetivo do desafio é classificar a categoria dos crimes ocorridos e a complexidade dessa tarefa reside na relativa simplicidade dos dados presentes no conjunto e também no desequilíbrio entre o total de instâncias para cada categoria (e.g., a categoria TRE tem apenas 6 instâncias no conjunto de treinamento, que possui mais de 800000 instâncias).

A seguir, apresentamos uma adaptação em português da descrição do desafio no Kaggle: "De 1934 a 1963, São Francisco ficou famosa por abrigar alguns dos criminosos mais notórios do mundo na inescapável ilha de Alcatraz. Hoje, a cidade é mais conhecida por sua cena tecnológica do que por seu passado criminal. No entanto, com o aumento da desigualdade de riqueza, escassez de moradias e a proliferação de dispositivos digitais caros sendo usados no BART para ir ao trabalho, não há escassez de crimes na cidade à beira da baía. Do bairro Sunset a SOMA, e da Marina ao Excelsior, o conjunto de dados desta competição fornece quase 12 anos de relatórios de crimes de todos os bairros de São Francisco. Dados o tempo e a localização, você deve prever a categoria de crime que ocorreu."

O restante deste artigo está organizado da seguinte forma: A Seção 2 apresenta uma fundamentação teórica necessária para compreensão do trabalho. Na Seção 3, é apresentado o desenvolvimento do código e problemas encontrados ao longo do caminho. Em seguida, na Seção 4 são apresentados os resultados obtidos e, por fim, a Seção 5 apresenta a conclusão e trabalhos futuros relacionados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo abordaremos os conceitos necessários para a compreensão do trabalho desenvolvido. A Seção 2.1 revisa conceitos de mineração de dados como identificação e tratamento de atributos e a Seção 2.2 revisa conceitos sobre regras de associação. Por sua vez, a Seção 2.3 trata de aprendizado de máquina e classificação. Por fim, a Seção 2.4 conlui o capítulo.

2.1 Pré-processamento

Pré-processamento refere-se à manipulação, filtração ou aumento de dados antes de sua análise e é um passo importante no processo de mineração de dados. O objetivo principal dessa etapa é garantir que os dados estejam em um formato adequado e contenham informações relevantes para a análise, melhorando a eficácia e desempenho dos modelos que usarão esses dados. Entre tarefas comumente realizadas no pré-processamento, podemos listar a limpeza dos dados, que identifica e corrige erros, valores ausentes ou inconsistentes, a normalização e transformação dos dados, para atender os requisitos do modelo, e, por fim, a seleção dos atributos mais interessantes para o problema em mão.

Os atributos representam as características ou propriedades que descrevem as instâncias do conjunto de dados. Em outras palavras, são as variáveis que contêm informações sobre os elementos estudados. Atributos podem ser classificados como: 1) contínuos, podendo assumir qualquer valor dentro de um intervalo (por exemplo, salário); 2) discretos, assumindo valores distintos e contáveis (por exemplo, idade); 3) categóricos, que representam categorias ou rótulos e podem ter uma ordem ou hierarquia natural (categóricos nominais, como cor) ou não (categóricos ordinais, como nível de educação); ou 4) binários, uma forma especial de atributos categóricos que têm apenas dois valores possíveis, geralmente representados por 0 ou 1, e podem ser simétricos, em que as duas categorias são igualmente válidas, sem ordem inerente entre elas (por exemplo, sexo), ou assimétricos, em que há uma ordem implícita entre as categorias (por exemplo, a presença de um efeito é mais importante que sua ausência).

Authors' address: Guilherme Augusto Anício Drummond do Nascimento, guilherme.drummond@aluno.ufop.edu.br; Henrique Dantas Pighini, henrique.pighini@aluno.ufop.edu.br, Universidade Federal de Ouro Preto, Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, Brasil, 35400-000.

ACM XXXX-XXXX/2024/2-ART
https://doi.org/XXXXXXX.XXXXXXX

2.2 Regras de associação

Regras de associação são utilizadas para descobrir relações interessantes entre variáveis em grandes conjuntos de dados. Uma regra de associação representa um padrão de relacionamento entre itens do domínio da aplicação que ocorre com uma determinada frequência na base de dados. A partir de um conjunto de transações de uma base de dados, tem-se um conjunto de regras na forma $X \rightarrow Y$, onde X é o antecedente da regra e Y é o consequente e ambos são conjuntos de itens do domínio da aplicação.

Regras de associação possuem índices que indicam sua relevância e validades, o suporte e a confiança, respectivamente. O fator de suporte de uma regra $X \rightarrow Y$ é definido pela porcentagem de transações que incluem todos os itens do conjunto $X \cup Y$ e representa a fração das transações que satisfazem tanto o antecedente quanto o consequente da regra. O fator de confiança de uma regra $X \rightarrow Y$ é definido pela porcentagem de transações que incluem os itens X e Y em relação a todas que incluem os itens de X e representa o grau de satisfatibilidade do consequente em relação às transações que incluem o antecedente.

2.2.1 Apriori. O processo de mineração das regras de associação pode ser dividido em duas fases. A primeira consiste em encontrar os conjuntos frequentes Z, tal que o suporte de Z seja maior ou igual ao suporte mínimo desejado, e a fase 2 consiste em, para conjunto frequente Z, identificar seus possíveis subconjuntos X e Y, tal que $Z = X \cup Y$ e a confiança da regra $X \rightarrow Y$ seja maior que a confiança mínima desejada.

O algoritmo Apriori procura os conjuntos frequentes Z, com o objetivo de reduzir o espaço de busca, considerando as seguintes propriedades: todo subconjunto de um conjunto frequente é frequente (se A, B, C é frequente, então A, B é frequente) e, pela contra-positiva, todo conjunto que contém um subconjunto não frequente também não é frequente (Se A, B não é frequente, então A, B, C não é frequente). Sua estratégia consiste em:

1 Calcular o suporte de todos os conjuntos de tamanho 1 e, em seguida, eliminar aqueles que não possuem o suporte mínimo.
2 Formar todos os possíveis conjuntos de tamanho 2 a partir daqueles de tamanho 1 resultados do passo anterior. Em seguida, eliminar os novos conjuntos que não possuam o suporte mínimo.
3 Repetir o procedimento anterior que que, no k-ésimo passo, nenhum novo conjunto de tamanho k, obtido a partir dos conjuntos de tamanho $k-1$, tenha suporte mínimo.

Para encontrar as regras de associação, aplicamos o seguinte procedimento:

- Para cada conjunto frequente t, gerar todos os seus subconjuntos não vazios.
- Para cada subconjunto s de t, tem-se a regra $s \rightarrow (t - s)$, se $\frac{suporte(t)}{suporte(s)} \geq$ confiança mínima

2.3 Aprendizado de máquina

O aprendizado de máquina (AM) é uma disciplina transformadora dentro da inteligência artificial (IA), capacitando sistemas computacionais a aprender padrões e tomar decisões sem serem explicitamente programados para tal. Essa abordagem tem aplicações em uma variedade de setores, desde diagnósticos médicos e finanças até reconhecimento de voz e veículos autônomos.

Algoritmo 1 Pseudocódigo do algoritmo Apriori

$F_1 \leftarrow$ conjuntos frequentes de tamanho 1
$k \leftarrow 1$
enquanto$(F_k \neq \emptyset)$:
  $k \leftarrow k + 1$
  $C_k \leftarrow$ candidatos de tamanho k gerados a partir de F_{k-1}
  para cada transação t da base de dados **faça:**
    Incrementar o contador associado a todo candidato C_k cujos itens pertençam a t
  $F_k \leftarrow \{c \in C_k | Suporte(c) \geq SuporteMinimo\}$
Resposta $F \leftarrow \bigcup_{i=1}^{k} F_i$

Aprendizado de máquina é um campo da inteligência artificial que se concentra no desenvolvimento de algoritmos capazes de aprender e melhorar sua performance ao longo do tempo. Ele se baseia na ideia de que os sistemas podem aprender com dados, identificar padrões e tomar decisões sem intervenção humana direta. Os principais conceitos incluem:

- **Modelos de aprendizado:** representações matemáticas que refletem o relacionamento entre variáveis em um conjunto de dados.
- **Treinamento e teste:** divisão do conjunto de dados em partes para treinar o modelo e avaliar sua performance.
- **Supervisionado e não supervisionado:** no aprendizado supervisionado, o modelo é treinado com pares de entrada e saída esperada. No não supervisionado, o modelo busca padrões sem rótulos explícitos.

Existem três tipos principais de aprendizado de máquina. O aprendizado supervisionado, onde o modelo é treinado em um conjunto de dados rotulado, onde a entrada e a saída desejada são conhecidas. O objetivo é fazer previsões ou classificações. O aprendizado não supervisionado, onde o modelo é treinado em um conjunto de dados sem rótulos, buscando padrões e estruturas subjacentes. Agrupamento e redução de dimensionalidade são exemplos. E, por fim, o aprendizado por reforço, onde o modelo aprende através da interação com um ambiente, recebendo recompensas ou penalidades com base em suas ações.

O aprendizado de máquina é aplicado em diversas áreas, como medicina, para diagnóstico de doenças e identificação de padrões em imagens médicas, finanças, para previsão de tendências de mercado e detecção de fraudes, e tecnologia, no reconhecimento de voz, em visão computacional e em carros autônomos.

2.3.1 Random Forest. Floresta aleatória (do inglês Random Forest) é um método de aprendizado em conjunto que se aplica a tarefas de classificação, regressão e outras, construindo diversas árvores de decisão durante o treinamento. No contexto de classificação, a saída do modelo é a classe mais escolhida entre as árvores, enquanto em tarefas de regressão, a previsão é a média das previsões das árvores individuais. Essa técnica corrige a propensão das árvores de decisão em se ajustarem demais ao conjunto de treinamento.

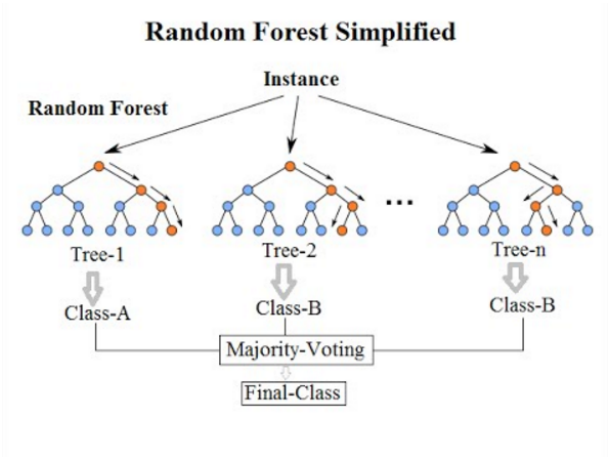

Fig. 1. Visualização simples de uma random forest, retirada de [Koehrsen 2017]

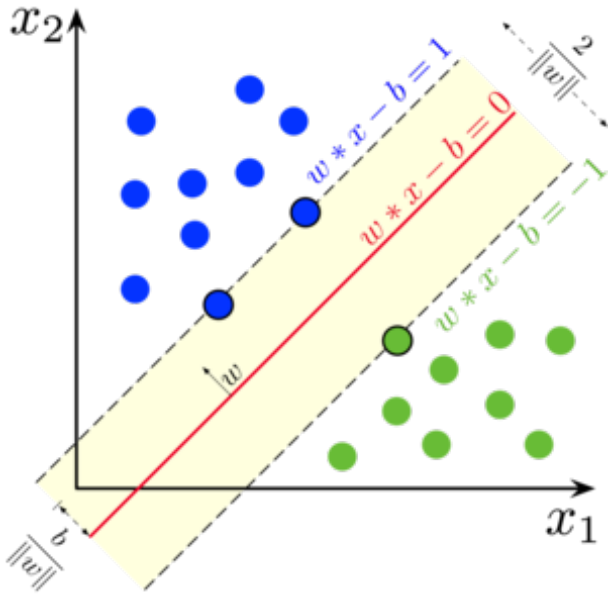

Fig. 2. Visualização simples de uma SVM, retirada de [Wikipedia contributors 2024]

Especificamente, quando as árvores são construídas com muita profundidade, elas podem aprender padrões excessivamente irregulares, resultando em baixo viés e alta variância. A floresta aleatório surge como uma solução, promovendo a média de várias árvores profundas treinadas em diferentes partes do mesmo conjunto de treinamento, visando a redução da variância. Essa abordagem implica em um ligeiro aumento no viés e alguma perda de interpretabilidade, mas, em geral, melhora significativamente o desempenho do modelo final. A Figura 1 apresenta um exemplo simplificado de uma random forest.

Embora as florestas aleatórias frequentemente superem uma única árvore de decisão em precisão, elas abrem mão da interpretabilidade inerente dessas árvores. As árvores de Decisão estão entre os modelos de aprendizado de máquina mais facilmente interpretáveis, juntamente com modelos lineares, baseados em regras e em atenção. Essa capacidade de interpretação é altamente valorizada, permitindo que os desenvolvedores confirmem que o modelo aprendeu informações realistas dos dados e permitindo que os usuários finais confiem nas decisões do modelo. Embora seja trivial seguir o caminho de decisão de uma única árvore, essa tarefa torna-se consideravelmente mais desafiadora com dezenas ou centenas de árvores. Para conciliar desempenho e interpretabilidade, algumas técnicas de compressão de modelo possibilitam transformar uma floresta aleatória em uma árvore de Decisão "renascida", mantendo a mesma função de decisão. Em situações onde os atributos preditivos estão linearmente correlacionados com a variável-alvo, o uso de uma floresta aleatória pode não resultar em ganhos significativos de precisão para o modelo base. Adicionalmente, em problemas que envolvem múltiplas variáveis categóricas, a floresta aleatória pode não conseguir aumentar a precisão do modelo base.

2.3.2 SVM. As Máquinas de Vetores de Suporte, ou SVM (do inglês Support Vector Machines), são uma poderosa técnica de aprendizado de máquina utilizada tanto para tarefas de classificação quanto para regressão. Essa abordagem é especialmente eficaz em espaços de alta dimensionalidade e é fundamentada na busca por um hiperplano de decisão otimizado para separar classes ou realizar previsões.

O principal objetivo das SVM é encontrar o hiperplano de decisão que melhor separa as instâncias pertencentes a diferentes classes em um espaço de características. Para isso, as SVM procuram maximizar a margem entre as instâncias das classes, definida como a distância entre o hiperplano e as instâncias mais próximas de cada classe, conhecidas como vetores de suporte. O hiperplano é um espaço de dimensão $n - 1$ em um espaço de n dimensões, onde n é o número de características do conjunto de dados. Em um problema de classificação binária, o hiperplano separa duas classes e é escolhido de maneira a maximizar a margem. A Figura 2 apresenta um exemplo simplificado de uma SVM.

A margem é a distância entre o hiperplano de decisão e os pontos mais próximos de cada classe. A SVM busca o hiperplano que maximiza essa margem, proporcionando uma maior robustez ao modelo. Os vetores de suporte são as instâncias do conjunto de dados que estão mais próximas do hiperplano de decisão. Eles desempenham um papel crucial na definição do hiperplano e na determinação da margem. A alteração ou remoção de vetores de suporte pode impactar significativamente a posição do hiperplano.

Em casos em que os dados não são linearmente separáveis, a SVM utiliza funções de *kernel* para mapear os dados para um espaço de maior dimensão, onde a separação linear é possível. Isso permite que as SVM lidem eficazmente com problemas complexos. O processo de aplicar uma função de *kernel* aos dados é chamado de *kernel*ização. Os tipos comuns de funções de *kernel* incluem o *kernel* linear, o *kernel* polinomial e o *kernel* radial (RBF/Gaussiano).

As SVM são amplamente utilizadas em problemas de classificação, como reconhecimento de imagem, detecção de spam, diagnóstico médico, entre outros. Além disso, podem ser aplicadas em tarefas

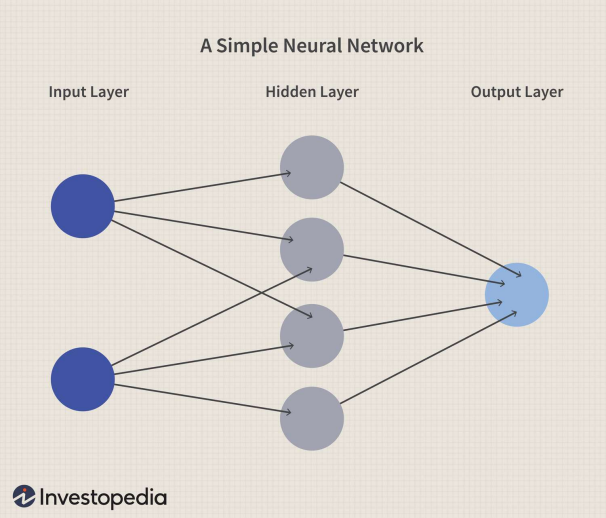

Fig. 3. Visualização simples de uma rede neural, retirada de [Chen 2023]

de regressão para prever valores contínuos. Elas são eficazes em espaços de alta dimensionalidade, podem lidar com conjuntos de dados complexos e são robustas em relação a *overfitting*, especialmente quando a margem é maximizada. No entanto, as SVM também têm desafios, como a sensibilidade à escala dos dados e a necessidade de ajustar cuidadosamente parâmetros, como a constante de regularização (C) e o tipo de *kernel*, para obter o melhor desempenho.

2.3.3 Redes neurais. Redes neurais são um paradigma de aprendizado de máquina inspirado na estrutura e no funcionamento do cérebro humano. Elas consistem em um conjunto interconectado de unidades básicas, chamadas neurônios artificiais, organizados em camadas. Cada neurônio recebe entradas, realiza cálculos ponderados e gera uma saída que pode ser usada como entrada para outros neurônios.

Cada neurônio artificial funciona como uma unidade de processamento. Eles recebem várias entradas, aplicam pesos a essas entradas, realizam uma soma ponderada e, em seguida, passam o resultado por uma função de ativação para gerar a saída e são organizados em camadas. A camada de entrada recebe os dados originais, a camada de saída produz as previsões ou resultados desejados, e entre elas, existem camadas intermediárias chamadas camadas ocultas. A profundidade da rede é determinada pelo número de camadas ocultas. A Figura 3 apresenta um exemplo simplificado de uma rede neural.

Cada conexão entre neurônios tem um peso associado. Esses pesos representam a força e a direção da influência que uma unidade exerce sobre a outra. O processo de treinamento da rede envolve ajustar esses pesos para otimizar o desempenho. Após a soma ponderada das entradas, é aplicada uma função de ativação para introduzir não linearidades na rede. Isso permite que a rede aprenda padrões complexos e relações não lineares nos dados. O treinamento de uma rede neural envolve a apresentação de dados de treinamento à rede, ajustando os pesos com base nas diferenças entre as previsões da rede e os resultados reais. O algoritmo de otimização, como o gradiente descendente, é frequentemente usado nesse processo.

Redes neurais são utilizadas em diversas aplicações, incluindo reconhecimento de imagem, processamento de linguagem natural, reconhecimento de voz, jogos, previsão de séries temporais, entre outros. Seu poder reside na capacidade de aprender representações automaticamente a partir dos dados. Porém, elas podem ser suscetíveis a *overfitting*, requerem grandes quantidades de dados para treinamento eficaz e podem ser computacionalmente intensivas. Além disso, questões éticas relacionadas à transparência e interpretabilidade são relevantes ao usar redes neurais em decisões críticas.

2.4 Conclusão

Neste capítulo apresentamos o referencial teórico deste trabalho: inicialmente revisamos os principais conceitos sobre mineração de dados. Na sequência, apresentamos o conceito de regras de associação e o algoritmo Apriori. Adicionalmente, apresentamos conceitos de aprendizado de máquina e os algoritmos de *RandomForest*, de *SVM* e de redes neurais.

3 DESENVOLVIMENTO

Neste capítulo abordaremos os passos necessários para a compreensão do trabalho desenvolvido. A Seção 3.1 apresenta o que foi desenvolvido em relação ao pré-processamento e transformação dos dados, a Seção 3.2 apresenta o que foi desenvolvido em relação a mineração de regras de associação e a Seção 3.3 apresenta estratégias para atingir o objetivo do desafio. Por fim, a Seção 3.4 conclui o capítulo.

3.1 Fase 1 - Pré-processamento e transformação

O conjunto de dados possui os seguintes atributos:

- Dates;
- Category (atributo alvo, presente apenas no conjunto de treinamento);
- Descript (presente apenas no conjunto de treinamento);
- DayOfWeek;
- PdDistrict;
- Resolution (presente apenas no conjunto de treinamento);
- Address;
- X;
- Y;
- Id (presente apenas no conjunto de teste).

Para a categorização de tipo de dados vamos usar as seguintes definições:

- Dados Contínuos: Podem assumir qualquer valor dentro de um intervalo, e há uma infinitude de valores possíveis.
- Dados Discretos: Representam valores distintos e contáveis, geralmente números inteiros, e não podem ser subdivididos em partes menores.
- Dados Categóricos: Representam categorias ou rótulos e pode ou não ter uma ordem ou hierarquia natural.
  - Dados Categóricos Nominais: Representam categorias sem uma ordem ou hierarquia intrínseca entre elas, sendo apenas rótulos descritivos.
  - Dados Categóricos Ordinais: Representam categorias com uma ordem ou hierarquia específica entre elas, permitindo

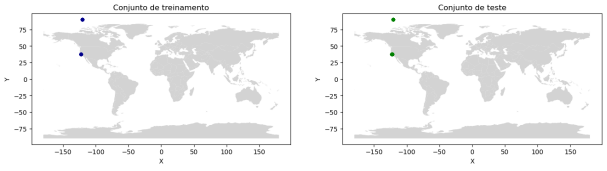

Fig. 4. Coordenadas plotadas em um mapa-múndi

a classificação ou ordenação com base em alguma característica subjacente.

- Dados Binários: Dados binários são uma forma especial de dados categóricos que têm apenas duas categorias possíveis, geralmente representadas como 0 e 1.
  - Simétricos: Quando não há ordem inerente entre as duas classes.
  - Assimétrico: Quando uma das classes tem mais importância do que a outra.

Dates representa a data em que o crime ocorreu, formatado como 'Y-m-d H:i:s'. Os conjuntos de treinamento e teste rotacionam toda semana, ou seja, semanas 1,3,5,7... são do conjunto de treinamento e semanas 2,4,6,8... são do conjunto de treinamento. Ele é um tipo dado discreto, no intervalo de '2003-01-01 00:01:00' a '2015-05-13 23:53:00'. Esse atributo não possui valores nulos.

Category representa as possíveis categorias que um crime pode ser classificado. É um tipo de dado categórico nominal e pode ser distribuído nos seguintes valores com suas respectivas populações, mostrado na Tabela 1. Category é o valor que o modelo tentará prever e não está presente no conjunto de testes.

Descript representa uma breve descrição expandida do crime. Está presente apenas no conjunto de treinamento e é um tipo de dado categórico nominal. DayOfWeek representa o dia da semana em que o crime ocorreu e é um tipo de dado categórico ordinal. PdDistrict representa o departamento de polícia que ficou encarregado do crime e é um tipo de dado categórico nominal. Resolution representa qual fim teve um crime, está presente apenas no conjunto de treinamento e é um tipo de dado categórico nominal. Address representa o endereço aproximado em que o crime ocorreu e é um tipo de dado categórico nominal. X e Y representam as coordenadas em que o crime aconteceu. É utilizado para identificar com precisão onde o crime foi registrado e é um tipo de dado contínuo.

Plotando os pontos em um mapa-múndi, como mostrado na Figura 4, podemos ver que alguns dos crimes estão localizados no Ártico. No conjunto de treinamento, 67 instâncias apresentam essas coordenadas errôneas e foram removidas do treinamento, enquanto no conjunto de teste, 76 instâncias apresentam essas coordenas errôneas. Todas as outras instâncias apresentam coordenas que se encontram dentro dos limites de São Francisco.

Como os atributos Descript e Resolution não se encontram no conjunto de teste, foram removidos do treinamento, pois não se mostrarão úteis para desenvolvimento e treinamento do modelo de classificação. O atributo Id, atribuído de forma única para cada crime observado, é um atributo que aparece apenas no conjunto de teste e será usado para criar o arquivo de envio final e, portanto, não faz parte do processo de predição.

Category	Total de instâncias
LARCENY/THEFT	174900
OTHER OFFENSES	126182
NON-CRIMINAL	92304
ASSAULT	76876
DRUG/NARCOTIC	53971
VEHICLE THEFT	53781
VANDALISM	44725
WARRANTS	42214
BURGLARY	36755
SUSPICIOUS OCC	31414
MISSING PERSON	25989
ROBBERY	23000
FRAUD	16679
FORGERY/COUNTERFEITING	10609
SECONDARY CODES	9985
WEAPON LAWS	8555
PROSTITUTION	7484
TRESPASS	7326
STOLEN PROPERTY	4540
SEX OFFENSES FORCIBLE	4388
DISORDERLY CONDUCT	4320
DRUNKENNESS	4280
RECOVERED VEHICLE	3138
KIDNAPPING	2341
DRIVING UNDER THE INFLUENCE	2268
RUNAWAY	1946
LIQUOR LAWS	1903
ARSON	1513
LOITERING	1225
EMBEZZLEMENT	1166
SUICIDE	508
FAMILY OFFENSES	491
BAD CHECKS	406
BRIBERY	289
EXTORTION	256
SEX OFFENSES NON FORCIBLE	148
GAMBLING	146
PORNOGRAPHY/OBSCENE MAT	22
TREA	6

Table 1. Lista das diferentes categorias e total de instâncias de cada categoria no conjunto de treinamento.

Os atributos foram alterados das seguintes maneiras: PdDistrict foi codificado em um intervalo de 0 a 9. Category foi codificado em um intervalo de 0 a 38. DayOfWeak foi codificado em um intervalo de 0 a 6. Address foi um atributo mais difícil de se decidir o que seria feito, no começo separamos as ruas e normalizamos, porém, percebemos que existiam conjuntos que tinham elementos parecidos ou eram subconjuntos um dos outros, então para minerar regras foi testado novas maneiras de separar ruas de blocos, e no fim acabamos descartando esse atributo pela sua complexidade e por termos coordenadas que nos retornam melhores resultados de localização do que nomes de ruas arbitrárias. As coordenadas X e Y

foram normalizadas, pois eram atributos numéricos que representam um ponto em um sistema de referência. Dates foi separado em novos atributos: Year, Month e Day, e a partir das horas também criamos o atributo Period que representa o período que aconteceu o crime: Manhã (06:00 - 12:00), Tarde (12:00 - 18:00), Noite (18:00 - 00:00) e Madrugada (00:00 - 06:00).

3.2 Fase 2 - Mineração de regras de associação

Para mineração das regras de associação foram usadas a biblioteca apyori[1] para mineração dos conjuntos frequentes e regras de associação e a biblioteca prefixspan[2] para mineração de padrões de sequências.

3.2.1 Mineração de regras de associação e padrões de sequência. O primeiro passo para poder minerar as regras de associação foi converter o banco de dados para um formato transicional. Esse passo consistiu em transformar cada instância i do banco de dados em uma lista de itens, no formato de "Nome do atributo j = valor de j na instância i". No caso do atributo Address, que pode ocorrer em um cruzamento de duas ruas, o atributo Address se repete, mas com valores diferentes.

Atributo	Valor
DayOfWeek	3
PdDistrict	4
Address	[OAK, LAGUNA]
X	5
Y	5
Year	2015
Month	5
Day	13
Period	Noite

Table 2. Exemplo de uma instância do banco já transformada

Example 3.1. A instância mostrada na Tabela 2 seria convertida na seguinte lista: ['DayOfWeek = 3', 'PdDistrict = 4', 'Address = OAK', 'Address = LAGUNA', 'X = 5', 'Y = 5', 'Year = 2015', 'Month = 5', 'Day = 13', 'Period = Noite', 'Category = WARRANTS']

Além de tentar encontrar regras de associação em todo o conjunto de dados, as instâncias foram agrupadas pelo ano, para procurar regras de associação dentro do período de um ano. Além disso, também foram geradas regras de associação agrupando pelo atributo PdDistrict.

Para a mineração de padrões de sequência foi usado apenas o atributo alvo Category. As instâncias foram agrupadas pelo ano em que o crime ocorreu.

3.3 Fase 3 - Fase final

Começando a fase 3 foi definido então nossa estratégia para entregar um modelo que retornasse o menor valor possível de loss definido da seguinte forma na página do desafio no Kaggle:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij})$$

onde N é o número de casos no conjunto de teste, M é o número de classes, log é o logaritmo natural, y_{ij} é 1 se o crime i está na classe j e 0 caso contrário, e p_{ij} é a probabilidade predita que o crime i pertence a classe j.

A submissão no Kaggle é feita por um conjunto de possíveis valores para cada entrada do conjunto de teste, indicando a probabilidade da instâncias pertencer a cada classe. Esse tipo de saída é feito normalmente por uma rede neural e, por isso, definimos 3 algoritmos de aprendizado de máquina para tentar solucionar o problema: 1) SVM, que foi apontado como o que teria o pior resultado pela sua simplicidade e por nosso problema ser complexo demais para ele; 2) Random Forest foi apontado como um modelo interessante de se testar; e 3) uma rede neural autoral encontrada em fóruns do Kaggle sobre o desafio [Duda 2020]. Todos os algoritmos implementados tiveram cross validation com 10 folds.

Para executar cada modelo foi necessário transformar nossos dados mais uma vez para satisfazer as entradas dos algoritmos. Foi utilizado o LabelEncoder da biblioteca scikit-learn para codificar todos os atributos não numéricos e, em seguida, todos os valores foram normalizados para o intervalo (0,1). Isso é interessante pois valores muito altos, como Year, podem acabar ganhando alguma importância para os modelos, o que não é desejado.

Também foi necessário diminuir o número de instâncias usadas, para realizar testes factíveis: com 800.000 instâncias, levamos quase 4 horas para rodar o SVM, que é o modelo mais simples. Então, reduzimos nosso banco de dados selecionando todas as instâncias de um ano. Porém, nossa submissão ao Kaggle foi feita utilizando um modelo com todas as 800.000 instâncias. Isso foi possível, pois a submissão ao Kaggle não passou por cross validation, então seu tempo de execução, mesmo que grande (1 hora), não foi tão custoso como quando estávamos fazendo testes.

Assim, temos aproximadamente 30.000 instâncias no ano de 2015 que foram usadas para o teste dos modelos. Foi feito, então, um split no banco de dados de 80/20: 80% do banco para treinamento e 20% para teste. Por fim, utilizamos o OneHotEncoder, também da biblioteca scikit-learn, para transformar o atributo alvo em um vetor binário que representa o valor de cada categoria.

3.4 Conclusão

Neste capítulo apresentamos os passos realizados para desenvolvimento do trabalho, começando pelo pré-processamento dos dados, seguido da mineração de regras de associações e padrões de sequências e, por fim, da criação do modelo final para predição das instâncias de teste.

4 RESULTADOS

Neste capítulo abordaremos os resultados de nossos testes e suas interpretações. Na Seção 4.1 vamos discutir sobre as estratégias que utilizamos e na Seção 4.2 vamos apresentar e abordar nossos resultados. Por fim, a Seção 4.3 conclui o capítulo.

[1]Disponível em https://pypi.org/project/apyori/
[2]Disponivel em https://pypi.org/project/prefixspan/

4.1 Avaliação experimental

Nosso plano para atacar o problema surgiu a partir de testes em uma rede neural pequena que não nos rendeu resultados plausíveis, assim, pesquisamos sobre nosso problema e traçamos um objetivo de testar com vários modelos e aplicar o conhecimento que vimos em sala de aula.

Assim, pesquisamos em fóruns que estavam discutindo sobre o desafio e encontramos um que apresentava o plano do autor para solucionar o problema e uma rede neural autoral. Escolhemos não utilizar das métricas que o mesmo utilizou, porém, como não tivemos sucesso em implementar uma rede neural satisfatória, utilizamos essa rede neural autoral no nosso trabalho.

Dessa forma, decidimos utilizar SVM e Random Forests, por serem algoritmos que vimos em sala de aula. Como o arquivo que deveríamos submeter ao Kaggle tinha o formato da saída de uma Neural Network, decidimos implementar uma também para facilitar o envio.

4.2 Análise dos resultados

4.2.1 SVM. Começando pelo SVM obtivemos um resultado que fugiu das nossas expectativas: acurácia média de 0.28, desvio padrão da acurácia de 0.00.

Para entender esse resultado nós estudamos o atributo alvo e percebemos que quase 25% do nosso banco de dados era composto por 2 categorias: (LARCENY/THEFT, OTHER OFFENSES). Esse comportamento mostrou que o SVM provavelmente conseguiu maximizar seu resultado apenas separando essas classes assim conseguindo acertar 1 em 4 instâncias do conjunto de testes.

4.2.2 Random Forests. Partindo para as Florestas Aleatórias, inicialmente acreditávamos que essa teria um resultado superior ao algoritmo SVM por ser um modelo mais complexo, porém, o resultado foi inferior: acurácia média de 0.25, desvio padrão da acurácia de 0.01. Como foi discutido no tópico do SVM isso pode ter acontecido pela distribuição não padrão das categorias no banco de dados.

Além disso, selecionar uma única classe é equivalente a atribuir uma probabilidade de 1 para essa classe e uma probabilidade de 0 para todas as outras classes. Se essa abordagem for adotada e a classe selecionada estiver incorreta, a métrica de avaliação do Kaggle atribuirá a penalidade máxima por observação. Por isso, optamos, no final, por não utilizar esse modelo para fazer a submissão no desafio.

4.2.3 Rede Neural. Por fim, testamos a rede neural implementada no artigo [Duda 2020].

Fizemos em nossos testes um plot para entender cada um dos folds do nosso crossover validation. Tivemos também um cálculo de loss para entendermos como estava o funcionamento das nossas redes, exemplificado na Figura 5.

Como pode ser visto na Figura 5 nossa rede neural aprendeu bem e reduziu o máximo que pode o Loss. Porém, a mesma ainda chegou em uma precisão de 0.28 com desvio padrão de 0.01. Assim, chegamos num melhor resultado de 0.28 de precisão.

4.2.4 Entrega ao Kaggle. Como foi instruído nós realizamos uma entrega ao Kaggle sem cross validation.

Para realizar a entrega o problema requisitou um arquivo .csv:

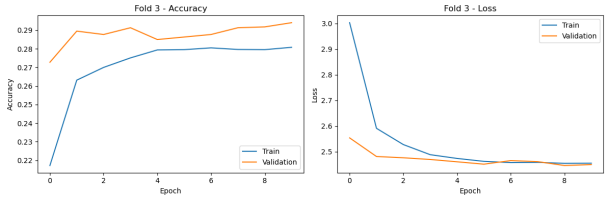

Fig. 5. Resultados plotados pelo Python

```
Id,ARSON,ASSAULT,BAD CHECKS,BRIBERY,BURGLARY,DISORDERLY CONDUCT,DRIVING UNDER THE
INFLUENCE,DRUG/NARCOTIC,DRUNKENNESS,EMBEZZLEMENT,EXTORTION,FAMILY
OFFENSES,FORGERY/COUNTERFEITING,FRAUD,GAMBLING,KIDNAPPING,LARCENY/THEFT,LIQUOR
LAWS,LOITERING,MISSING PERSON,NON-CRIMINAL,OTHER OFFENSES,PORNOGRAPHY/OBSCENE
MAT,PROSTITUTION,RECOVERED VEHICLE,ROBBERY,RUNAWAY,SECONDARY CODES,SEX OFFENSES
FORCIBLE,SEX OFFENSES NON FORCIBLE,STOLEN PROPERTY,SUICIDE,SUSPICIOUS
OCC,TREA,TRESPASS,VANDALISM,VEHICLE THEFT,WARRANTS,WEAPON LAWS
0,0.9,0.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
...
etc.
```

Fig. 6. Template de entregas Kaggle

Então utilizando a mesma rede neural, porém, com o conjunto inteiro do banco de dados, conseguimos um resultado pior: loss de 2.62, acurácia de 0.22.

E é interessante ver que o gráfico de Loss e acurácia demonstrou resultados flutuantes.

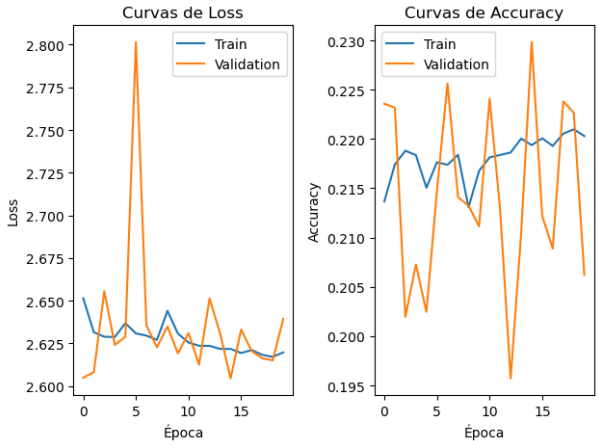

Fig. 7. Resultados do Kaggle

4.3 Conclusão

Neste capítulo apresentamos os resultados que obtivemos durante o desenvolvimento e testes, além de analisá-los. Também foi apresentado e analisado o resultado final obtido ao submeter nossa resposta ao desafio no Kaggle.

5 CONCLUSÃO

Este trabalho de mineração de dados sobre o desafio de classificação de crimes em São Francisco revelou-se desafiador devido à nossa falta de experiência e conhecimento abrangente em todos os

métodos empregados. Ao nos depararmos com as complexidades inerentes à análise de dados de crimes, ficou evidente que a ausência de experiência prejudicou nossa capacidade de explorar plenamente os potenciais de cada técnica.

Particularmente, a carência de familiaridade com os métodos utilizados limitou nossa habilidade de otimizar parâmetros e selecionar abordagens mais eficazes. Além disso, a falta de experiência foi um obstáculo para a criação de novos atributos significativos que poderiam ter enriquecido o processo de predição. A ausência de insights inovadores impactou diretamente a capacidade do modelo em capturar nuances importantes nos dados e, consequentemente, afetou a qualidade das previsões.

Apesar dos desafios enfrentados, este trabalho serviu como uma valiosa oportunidade de aprendizado, destacando a importância do aprimoramento contínuo de habilidades e conhecimentos em mineração de dados. Reconhecemos a necessidade de aprofundar nossa compreensão dos métodos utilizados, bem como de explorar estratégias mais avançadas para a criação de atributos relevantes.

Este processo de análise de dados não apenas ressaltou as dificuldades inerentes à falta de experiência, mas também motivou-nos a buscar aprimorar constantemente nossas habilidades, explorar novas técnicas e abordagens, e, assim, superar desafios futuros de forma mais eficaz.

REFERENCES

James Chen. 2023. What Is a Neural Network? — investopedia.com. https://www.investopedia.com/terms/n/neuralnetwork.asp. [Accessed 03-02-2024].

Scott Duda. 2020. San Francisco Crime Classification — scottmduda.medium.com. https://scottmduda.medium.com/san-francisco-crime-classification-9d5a1c4d7cfd. [Accessed 03-02-2024].

Wendy Kan. 2015. San Francisco Crime Classification. https://kaggle.com/competitions/sf-crime

Will Koehrsen. 2017. Random Forest Simple Explanation — williamkoehrsen.medium.com. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d. [Accessed 03-02-2024].

Wikipedia contributors. 2024. Support vector machine — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=1198920933. [Online; accessed 3-February-2024].