

Relatório de Análise de Dados

Membros do grupo:

Bruno Marques Bastos
Bruno Reck Gambim
Filipe Dornelles Gonçalves
Thales Junqueira Albergaria Moraes Perez
Victor Elitt Carvalho

1. Identificação do Problema e Coleta de Dados

Identificação do Problema:

Nossa primeira ideia era tentar cruzar informações relacionadas às chuvas em Porto Alegre e ao nível do Lago Guaíba para tentar simular, mesmo que dependente de diversos outros fatores, quando as cheias podem ocorrer. No entanto, pela dificuldade de se obter os dados necessários para resolver esse problema e tendo em vista o curto prazo para o desenvolvimento do trabalho, resolvemos ir para um problema mais simples. Optamos por escolher um problema que iríamos conseguir fazer com os dados reais do nosso estado ao invés de tentar buscar um problema mais complexo com um conjunto de dados pronto da internet.

O problema que escolhemos está relacionado com a predição de chuva. Através dos dados do dia de hoje, devemos prever se vai ou não chover amanhã. Consideramos que choveu em um dia caso tenha ocorrido uma precipitação maior que 1mm.

Descrição da Coleta de Dados:

Todos os dados utilizados foram coletados através do Instituto Nacional de Meteorologia (INMET). O INMET possui uma base de dados com diversos dados meteorológicos, como pressão atmosférica, temperatura, direção e velocidade do vento, precipitação, etc, que são coletados de hora em hora. Nessa etapa de coleta, buscamos todos os dados disponibilizados, desde o ano de 2001 até o ano de 2024, filtrando pelos dados coletados no estado do Rio Grande do Sul. Para os anos iniciais, existem dados apenas de 5 cidades do estado, mas esse número aumentou com o passar dos anos.

Os dados do INMET estão separados em arquivos no formato csv, onde cada arquivo contém os dados coletados em uma estação de coleta e em um ano específico. As primeiras linhas de cada arquivo contém alguns dados globais, como latitude, longitude, cidade, estado, código da estação de coleta, etc. Depois dessa parte, vem o restante dos dados, com cada linha contendo os dados coletados em uma hora específica de um dia específico do ano.

Nosso maior trabalho nessa etapa foi juntar esses dados em um único arquivo csv, agrupando as linhas pelo dia. Criamos colunas extras para poder juntar todos os arquivos em um. Essas novas colunas correspondem a cidade, a longitude, a latitude e ao código da estação de coleta. Já existia uma coluna de data com o ano da coleta, então não foi necessário criar uma coluna nova para isso. Para agrupar as linhas correspondentes às várias horas de um dia em apenas uma linha correspondente a um dia, usamos lógicas diferentes dependendo dos dados das colunas. Para as colunas com os dados relacionados ao vento, separamos cada uma delas em 4 novas colunas, com o valor dela às 0 horas, às 6 horas, às 12 horas e às 18 horas. Fizemos esse tratamento especial pois uma das colunas do vento diz respeito a direção em um determinado horário, então achamos que os dados poderiam perder o significado caso a gente apenas agrupasse todos eles em apenas uma coluna. Para a coluna de precipitação total,

somamos o valor de todas as horas. Para as colunas com valores do mínimo de alguma medida, calculamos o mínimo de todas as horas do dia. Tratamos o máximo de forma análoga. Por fim, para as outras colunas, calculamos a média entre as horas.

Algumas medidas de algumas horas não estavam presentes nos dados ou possuíam o valor ‘-9999’, possivelmente, indicando um erro. Apenas desconsideramos esses valores na hora de agrupar as linhas. Caso todas as horas de uma coluna específica de um determinado dia tivessem valores faltantes ou o valor de ‘-9999’, nós atribuímos o valor especial ‘NA’ da biblioteca do pandas para essa coluna desse dia. Em etapas futuras, nós fizemos o tratamento dos dados com valores ‘NA’ que adicionamos nessa etapa de agrupamento.

Além disso, antes de agrupar os dados, os valores de algumas colunas de mínimo e máximo eram relativos a hora anterior da linha a qual estavam posicionados. Fizemos um tratamento para reposicionar os valores dessas colunas para as linhas correspondentes a hora em que eles foram coletados.

Por fim, criamos uma nova coluna correspondente ao valor alvo do problema que escolhemos, isto é, uma coluna que indica se vai ou não chover no próximo dia. Para identificar se choveu ou não no próximo dia, verificamos se o valor da precipitação no próximo dia é maior que 1mm.

O script usado nessa parte está contido no arquivo “p.py”.

Conclusões:

Essa etapa foi relativamente desafiadora de ser realizada. Precisamos pensar em um tema relacionado às enchentes, encontrar dados para resolver esse problema além de não exagerar na complexidade e escolher um problema que não seja possível de realizar no prazo. Tendo em vista essas dificuldades, acreditamos que o problema escolhido satisfaz bem os requisitos propostos.

2. Análise Exploratória dos Dados

Análise Inicial:

Após agrupar os dados em um único arquivo, o conjunto de dados ficou com os seguintes atributos: precipitação total, vai chover amanhã, pressão média, pressão máxima, pressão mínima, radiação global, temperatura média, temperatura orvalho média, temperatura máxima, temperatura mínima, temperatura orvalho máxima, temperatura orvalho mínima, umidade máxima, umidade mínima, umidade média, cidade, código, latitude, longitude, data, direção do vento às 0 horas, direção do vento às 6 horas, direção do vento às 12 horas, direção do vento às 18 horas, rajada máxima de vento às 0 horas, rajada máxima de vento às 6 horas, rajada máxima de vento às 12 horas, rajada máxima de vento às 18 horas, velocidade média do vento às 0 horas, velocidade média do vento às 6 horas, velocidade média do vento às 12 horas e velocidade média do vento às 18 horas.

Optamos por remover as colunas cidade e código pois elas podem ser representadas pela latitude e longitude do ponto de coleta. Optamos por manter a latitude e a longitude ao invés de uma dessas duas informações, pois, além de evitar etapas de transformação de atributos categóricos em atributos numéricos, a latitude e a longitude trazem informação da posição dos pontos de coleta em relação aos outros.

Por fim, separamos a data em 3 atributos numéricos: 1 atributo para o dia, 1 atributo para o mês e 1 atributo para o ano.

Valores Faltantes:

A Figura 1 mostra um gráfico com os dias com valores faltantes para cada ano. A coluna azul são os dias com pelo menos um atributo faltante, a coluna vermelha são os dias em que todos os atributos são faltantes e a coluna amarela os dias onde pelo menos o atributo alvo é faltante.

Para lidar com esse problema, primeiramente, optamos por remover os dias em que pelo menos o atributo alvo era faltante. Fizemos essa escolha por conta da importância do atributo alvo no processo de predição, além de que já teríamos que remover uma boa parte desses dias tendo em vista que eles possuíam todos os seus atributos como faltantes. É importante destacar que o conjunto dos dias que possuem todos os atributos como faltantes está contido no conjunto dos dias que possuem o atributo alvo como faltante e, portanto, ao descartar o segundo conjunto, também descartamos o primeiro.

Para os dias com valores faltantes que restaram no conjunto de dados, definimos o valor deles como a média dos valores do seu respectivo atributo no conjunto de dados.

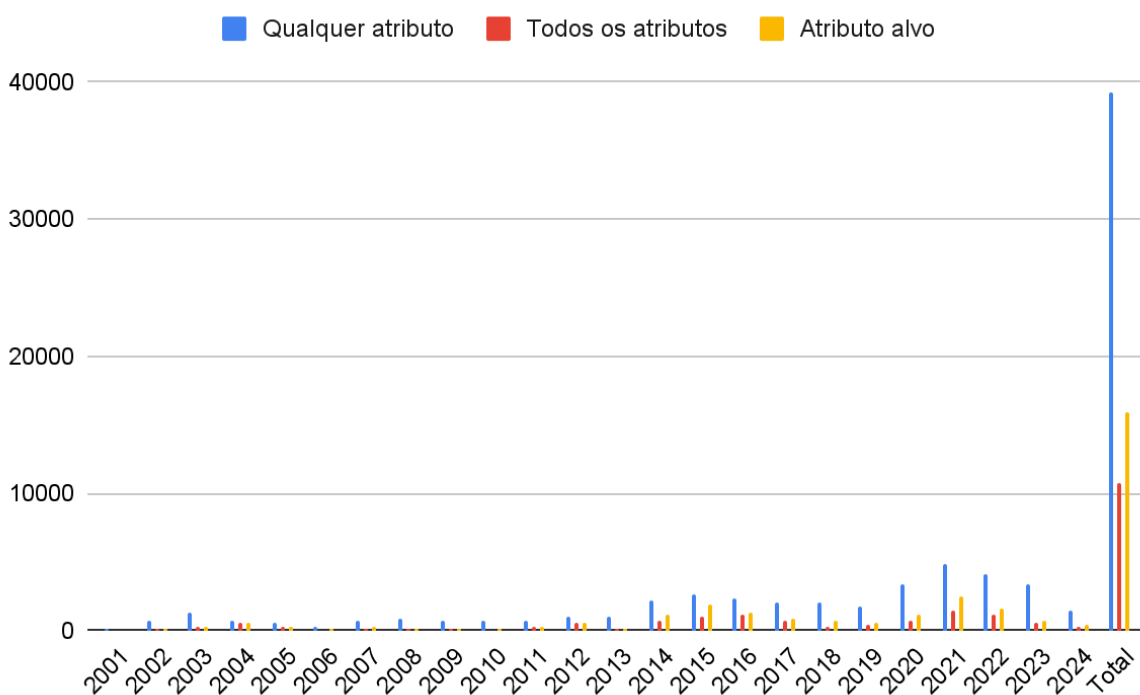


Figura 1: Gráfico dos dias com valores faltantes para cada ano.

Normalização:

Vários atributos do conjunto de dados possuem escalas diferentes, por exemplo, a umidade varia de 0 até 100 enquanto a direção do vento varia de 0 até 360. Portanto, é necessário fazer uma normalização dos dados.

Nessa etapa de normalização e em etapas futuras, vamos testar abordagens diferentes para resolver um problema de pré-processamento em alguns algoritmos específicos para avaliar essas abordagens. Para realizar esses testes, vamos usar os seguintes modelos da biblioteca do Scikit Learn:

- Um MLP com 1 camada de 10 neurônios, seguida por 2 camadas de 20 neurônios, seguida por uma camada final de 10 neurônios. Função de ativação ReLU e um máximo de 250 iterações.
- Uma floresta aleatória com profundidade máxima de 15 e 100 estimadores. Foi ativada uma flag para lidar com classes desbalanceadas.
- O algoritmo AdaBoost com 100 estimadores e o algoritmo SAMME. Os estimadores base usados são árvores de decisão com profundidade máxima de 5. Essas árvores de decisão possuem uma flag ativada para lidar com classes desbalanceadas.
- O algoritmo de regressão logística com um máximo de 5000 iterações e uma flag para lidar com classes desbalanceadas.

Outra observação geral é que esses gráficos de teste sempre possuem colunas base relativas a não usar nenhuma das alternativas propostas. Essa coluna base é executada com base no melhor conjunto de dados encontrado até o momento pelo pré-processamento. Como sempre vamos comparar o desempenho com uma base, podemos estimar se alguma das alternativas propostas conseguiu melhorar o nosso processo de pré-processamento. Nessa etapa e nas etapas futuras, vamos sempre informar qual das alternativas vai ser usada nas próximas etapas ou se nenhuma delas vai ser usada. No caso da etapa de normalização, essas colunas base são relativas aos dados após lidar com os valores faltantes.

Um ponto final relativo aos testes é que vamos priorizar uma melhora no Recall e na métrica F1 em detrimento das outras métricas. Mais para frente no relatório, explicaremos melhor essa escolha.

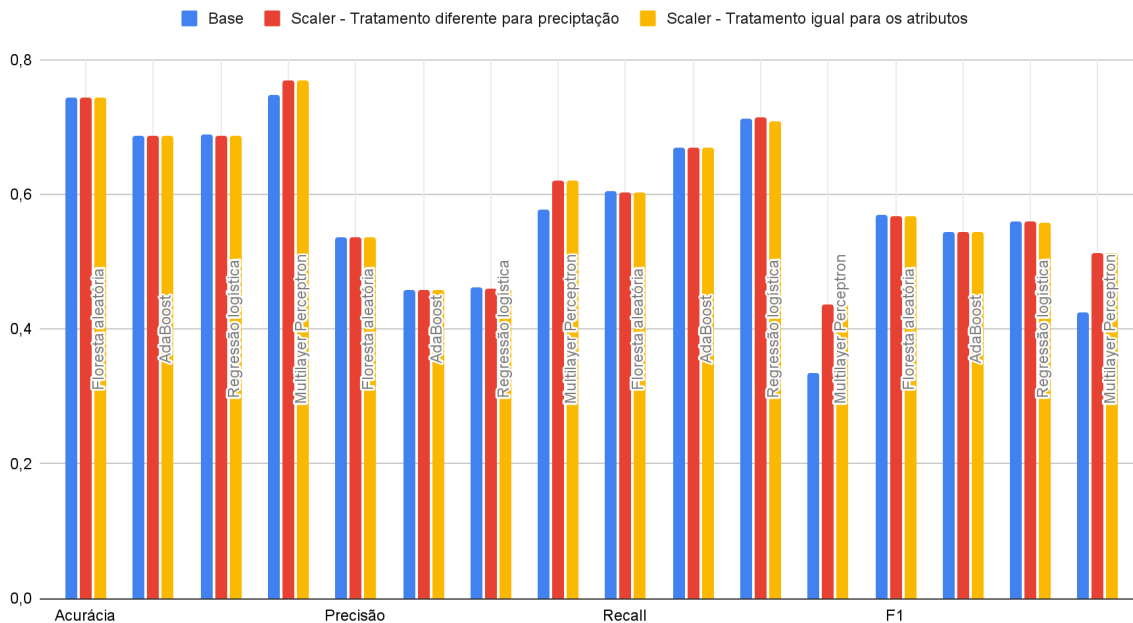


Figura 2: Gráfico com os resultados da execução dos algoritmos aplicando ou não as alternativas de normalização de atributos.

Nossa primeira tentativa de normalização foi simplesmente usando o “RobustScaler” do scikit learn. Porém, mesmo após essa normalização, a variação entre o valor máximo e o mínimo da precipitação total era em torno de 50 vezes maior que o de algumas colunas. O valor absoluto da média da precipitação total também era bem discrepante. Esse tipo de problema ocorreu possivelmente pois a maioria esmagadora dos valores da precipitação total é 0mm.

Para melhorar esse processo, normalizamos os valores da precipitação total usando uma lógica diferente dos outros atributos. Continuamos usando “RobustScaler” nos dois casos, mas mudamos o quantile_range usado na precipitação total para 6.5% e 93.5%. Esses valores foram encontrados através de testes. Com isso, resolvemos essa discrepância nos valores pós normalização.

O gráfico da Figura 2 mostra os testes usando cada um dos dois métodos ou não usando nenhum. As colunas azuis são relativas a não aplicar a normalização, as colunas amarelas a usar o “RobustScaler” de forma igual em todos os atributos e as colunas vermelhas a tratar a precipitação total de forma diferente. Podemos ver uma grande melhora no recall e na métrica F1 no MLP com a aplicação de qualquer uma das métricas. Porém, a normalização não teve um impacto tão grande nos outros modelos. Não é possível notar diferença significativa entre as abordagens de normalização através gráfico, mas a abordagem complexa é melhor na terceira casa decimal de algumas métricas em alguns algoritmos. Optamos por usar a abordagem que trata a precipitação total de forma diferente para os testes das próximas etapas.

Outliers:

Primeiramente, as colunas relacionadas à data não possuem outliers, pois, se elas tivessem, teria ocorrido um erro na etapa inicial, onde a data foi convertida para dia, mês e ano. Para buscar outliers nas colunas de latitude e longitude, comparamos se todas as latitudes e longitudes de cada código são iguais. Não encontramos nenhum outlier nessas duas colunas.

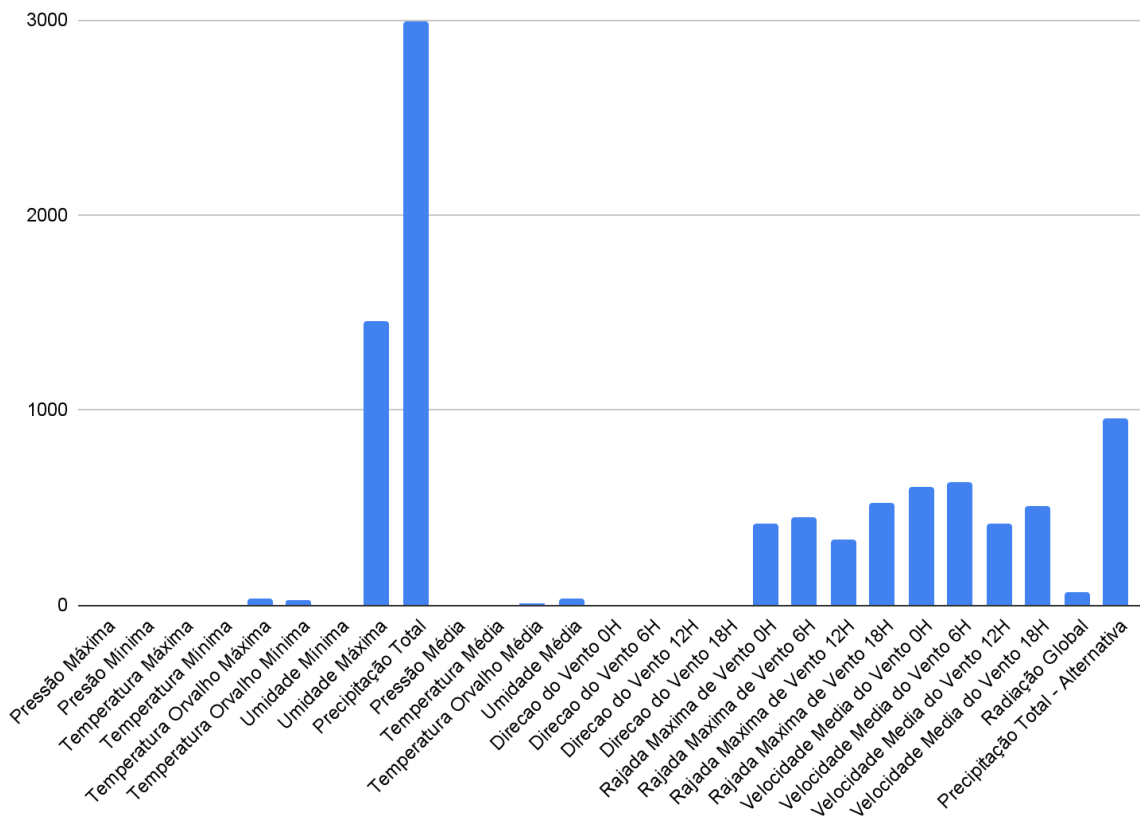


Figura 3: Gráfico do número de valores de cada atributo que tiveram z-score maior que 4.

Fizemos uma análise das colunas usando o z-score. Inicialmente comparamos todas as colunas usando uma mesma estratégia. De maneira semelhante à normalização, os valores da precipitação total ficaram muito maiores que os outros. Portanto, criamos uma abordagem alternativa, usando apenas os valores diferentes de 0 para analisar a precipitação total. A Figura 3 mostra um gráfico com o número de valores de cada atributo que tiveram z-score maior que 4. Podemos notar que a abordagem alternativa deixou o número de ocorrências para o atributo precipitação total menos destoante dos outros.

Usamos essa ideia do z-score para categorizar e tratar os outliers. Criamos duas abordagens diferentes:

- A primeira usa todos os valores de uma coluna para calcular a média e desvio padrão usados no z-score, considera os elementos com o z-score maior que X como outliers e substitui eles pela média dos valores não outliers do atributo.
- A segunda alternativa trata a precipitação de forma diferente, considerando apenas valores diferentes de 0 na hora da análise, tanto para o cálculo do z-score como para calcular a média que vai ser usada no lugar dos outliers.

A Figura 4 mostra os resultados dos testes considerando atributos com z-score maior que 4 como outliers. Lembrando que os testes desta etapa foram feitos usando o conjunto de dados de saída da etapa de normalização. Podemos notar que o recall e métrica F1 pioraram no MLP após aplicar os métodos e que o recall melhorou suavemente no adaboost usando o método mais complexo. Também podemos notar que o tratamento simples possui desempenho pior ou parecido com o do tratamento complexo em quase todos os casos. Considerando o desempenho significativamente inferior no MLP, optamos por não aplicar nenhum método para lidar com outliers nas próximas etapas.

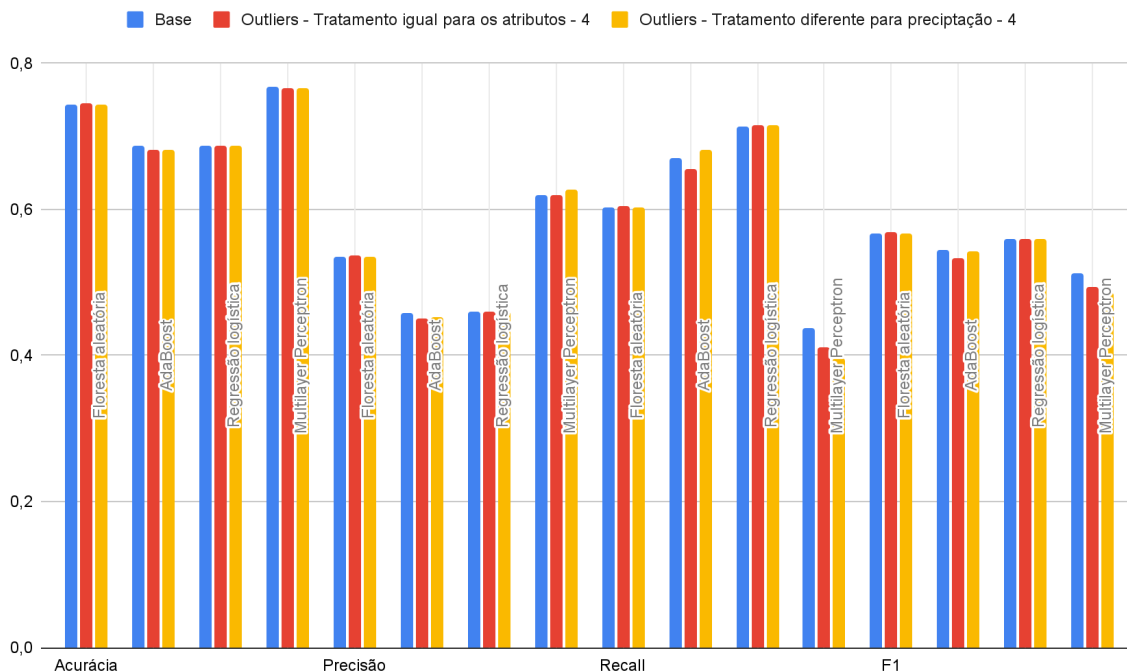


Figura 4: Gráfico com os resultados da execução dos algoritmos aplicando ou não as abordagens para lidar com outliers.

Uma observação sobre esse processo é que testamos outros valores para o z-score mas decidimos não colocar eles no relatório visto que eles possuíam um comportamento

parecido com o do z-score igual a 4, inferior ao do z-score igual a 4 ou possuíam um desempenho pior no adaboost em troca de aumentar o desempenho no MLP para algo que ainda era pior que a versão base.

Para entender melhor o motivo pelo qual não houve uma melhoria visível ao remover os outliers usando essa estratégia, a gente analisou os valores máximos e mínimos dos atributos que a técnica do z-score apontou com um maior número de outliers. As colunas de velocidade do vento não aparentam ter valores fisicamente absurdos como máximos e mínimos, com mínimos maiores que 0 m/s e máximos menores que 45m/s. O mesmo vale para a coluna de umidade máxima, com mínimo de 0% e máximo de 100%. Nem mesmo a precipitação total, que possui um mínimo de 0mm e um máximo de 238.6mm, o que não é tão absurdo considerando que pegamos dados dos períodos das enchentes. Fizemos uma análise parecida para as colunas restantes e não encontramos valores máximos e mínimos absurdos em nenhuma delas. Portanto, possivelmente o método que usamos está considerando valores corretos do conjunto de dados como outliers. Outro ponto a ser considerado é que, dada a natureza do problema, onde buscamos detectar a ocorrência de uma classe minoritária, talvez esses valores extremos do conjunto de dados, que o método do z-score está removendo, estão diretamente relacionados à ocorrência dessa classe minoritária.

Dados desbalanceados:

O problema possui classes alvo desbalanceadas, com 173371 valores da classe negativa e apenas 67081 valores da classe positiva. Testamos diversos algoritmos da biblioteca imbalanced-learn para tentar balancear as classes. A Figura 5 mostra um gráfico com testes após a execução desses algoritmos. O número X ao lado do SMOTE e Borderline SMOTE quer dizer que o algoritmo foi aplicado com o parâmetro k_neighbors igual a X.

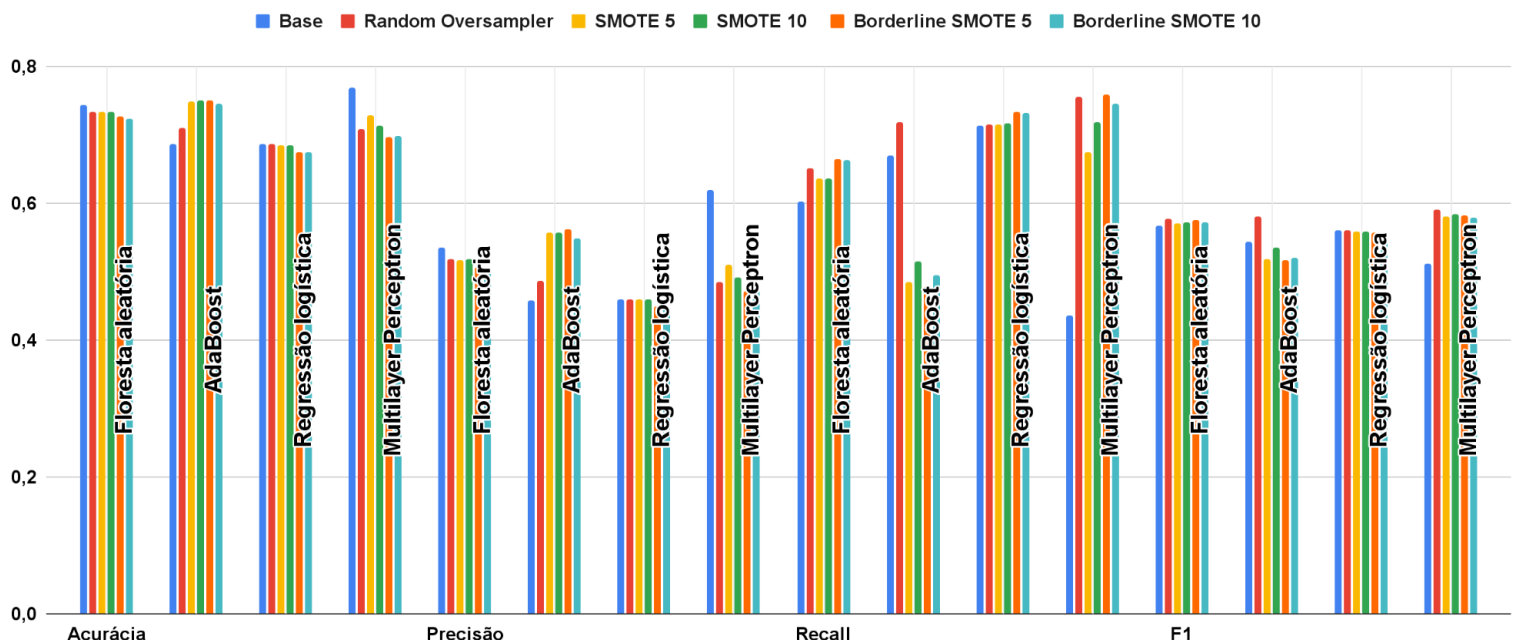


Figura 5: Gráfico com os resultados da execução dos algoritmos aplicando ou não as abordagens para lidar com o desbalanceamento de dados.

Primeiramente, podemos notar que, com exceção do AdaBoost, a acurácia diminuiu com a utilização dos métodos para lidar com o desbalanceamento. Também podemos notar que, no geral, o Random Oversampler teve o melhor desempenho considerando as métricas

F1 e o recall. Também podemos notar que, com exceção do AdaBoost, o Borderline SMOTE 5 conseguiu os melhores valores para o recall. Optamos manter o Random Oversampler nas próximas etapas por causa do seu desempenho geral entre os algoritmos para o Recall e para a métrica F1.

PCA:

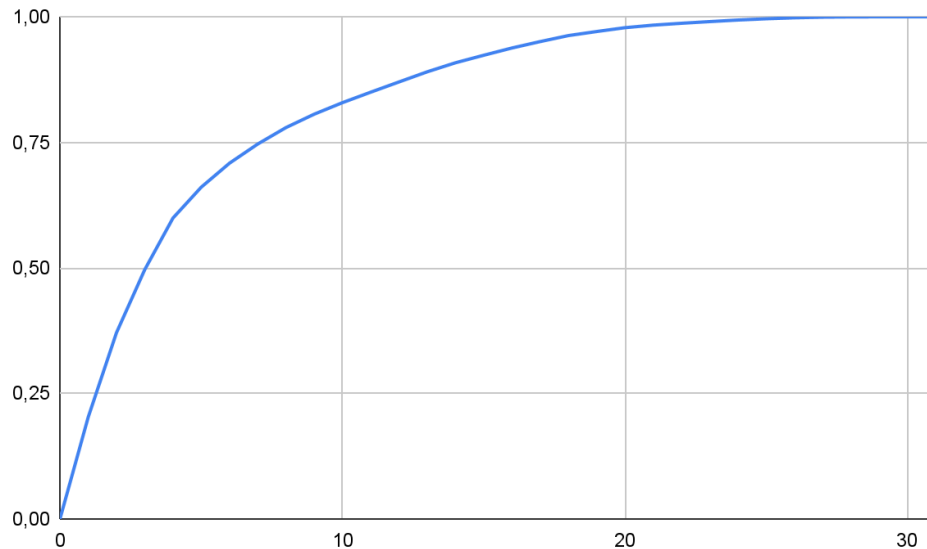


Figura 6: Gráfico com a variância explicada cumulativa.

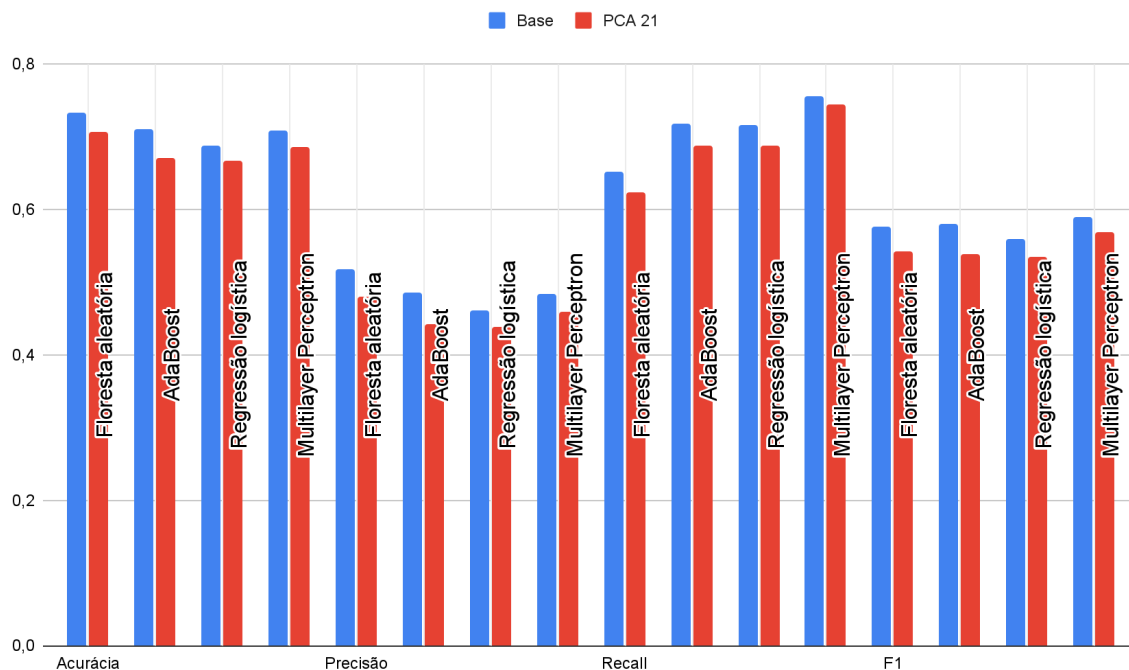


Figura 7: Gráfico com os resultados da execução dos algoritmos aplicando ou não o PCA.

Tentamos aplicar a transformação de atributos usando o PCA. A Figura 6 possui um gráfico com a variância explicada cumulativa. Usamos esse gráfico para escolher o número de

componentes a ser usado pelo PCA. Escolhemos o número 21, pois é o número no qual a variância explicada passa de 98%. A Figura 7 possui um gráfico com os testes da execução do PCA com 21 componentes. Podemos ver que o uso do PCA piorou o desempenho em todos os casos. Portanto, optamos por não usar o PCA nas próximas etapas.

Seleção de Features:

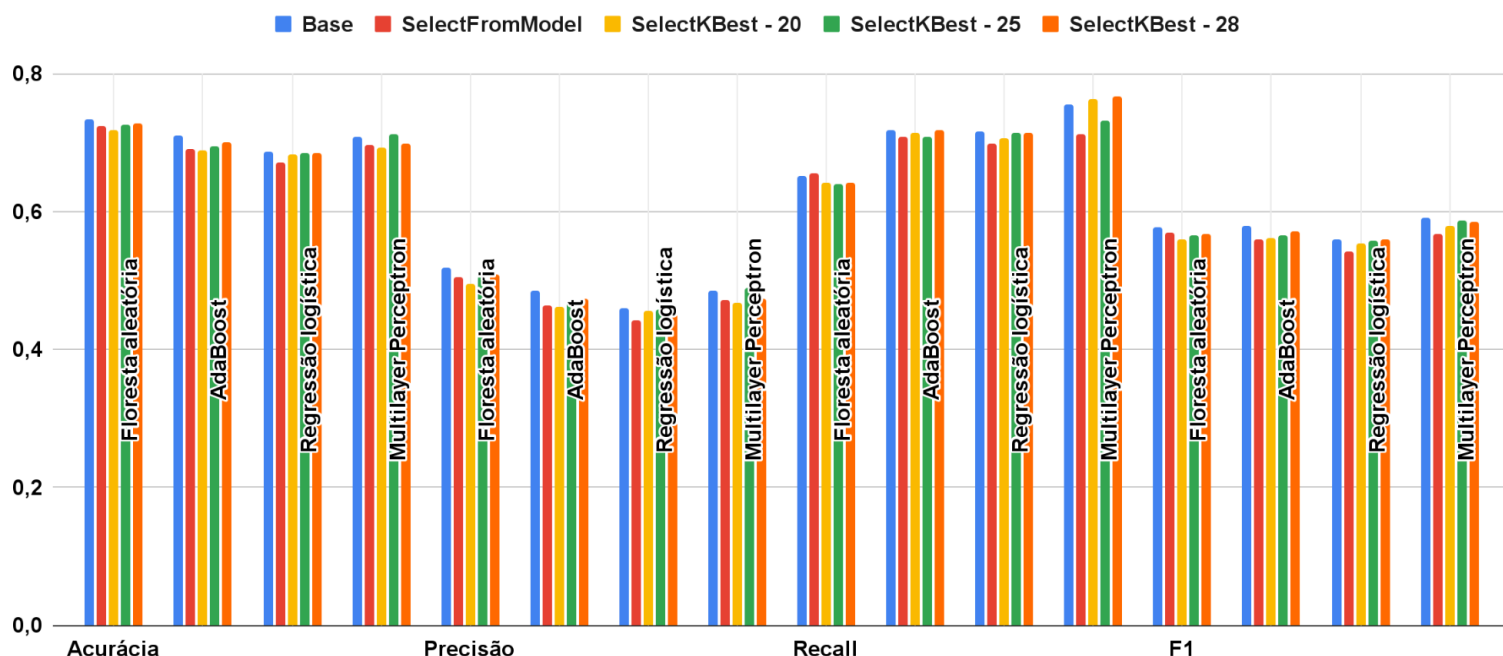


Figura 8: Gráfico com os resultados da execução dos algoritmos aplicando ou não a seleção de features.

Realizamos diversos testes com os algoritmos de seleção de features do scikit learn. A Figura 8 possui o resultado desses testes. O algoritmo SelectFromModel foi usado com uma floresta aleatória com 100 estimadores e profundidade máxima igual a 20. O número X ao lado de SelectKBest significa que o algoritmo foi executado com o parâmetro k igual a X. Podemos ver que, em quase todos os casos, os algoritmos de seleção de feature pioraram a precisão, a métrica F1 e a acurácia para todos os modelos. Teve um ganho no recall na floresta aleatória com o uso do SelectFromModel e um ganho no MLP com o SelectKBest com k igual a 20 ou 28. Tendo em vista o desempenho ruim na maioria dos casos, optamos por não usar seleção de features nas próximas etapas.

Conclusões:

Um primeiro ponto a ser notado é que nossas abordagens para remover os outliers não funcionaram muito bem junto com os modelos usados. Talvez seja interessante tentar abordagens mais complexas futuramente.

Além disso, em algumas etapas do pré-processamento, fizemos algumas escolhas de parâmetros. Um exemplo disso é o caso da normalização, onde escolhemos o quantile_range usado pelo algoritmo. No futuro, quando formos otimizar os modelos para resolver o problema, talvez faça sentido fazer uma otimização desses parâmetros também.

3. Definição da Abordagem, Métricas e Métodos de Avaliação

Abordagem Definida:

O problema que escolhermos resolver é um problema de aprendizado supervisionado de classificação. Ele é um problema de aprendizado supervisionado pois temos um atributo alvo que tentamos prever com base nos outros atributos. Ele é um problema de classificação pois o atributo alvo é categórico.

Métricas Utilizadas:

Vamos usar a acurácia, precisão, recall e a métrica F1 em conjunto para avaliar os modelos, dando uma prioridade maior ao recall e a métrica F1. A acurácia sozinha não pode ser usada por conta do desbalanceamento das classes dos dados. Como nosso problema está relacionado a prever se vai chover ou não, consideramos o caso em que chove e ele não avisa como pior do que o caso em que ele avisa e não chove. Então, vamos priorizar diminuir os falsos negativos do que os falsos positivos e, portanto, vamos priorizar o recall sobre a precisão. Por fim, não podemos usar o recall sozinho, pois um algoritmo que chutasse que iria chover todos os dias teria 100% de recall.

Métodos de Avaliação:

Como temos um conjunto relativamente grande de dados de testes, optamos por usar o método holdout com uma divisão de 80% dos dados para treino e 20% para teste nesta etapa de pré-processamento e de spot-checking para comparar soluções alternativas. Vale a pena observar que toda a etapa de pré-processamento foi realizada após os dados serem separados em treino e teste para evitar vazamento dos dados de teste para treino.

Em etapas futuras, pretendemos usar 5-fold cross validation para testar o desempenho dos modelos e realizar comparações entre eles.

Conclusões:

Essa etapa foi extremamente importante na realização das outras etapas. Definir o problema na árvore de aprendizado de máquina ajudou a gente a procurar os algoritmos para realizar os testes e uma definição inteligente das métricas ajudou a gente a avaliar estes testes.

Além disso, foi fundamental entender melhor as características dos dados e do problema para conseguir escolher as métricas que utilizamos na avaliação bem como a prioridade dessas métricas.

4. Spot-Checking

Descrição do Spot-Checking:

Usamos os seguintes modelos na etapa do Spot-Checking:

- MLP com 1 camada de 10 neurônios, seguida por 2 camadas de 20 neurônios, seguida por uma camada final de 10 neurônios. Função de ativação ReLU e um máximo de 250 iterações;
- Floresta Aleatória com profundidade máxima de 15 e 100 estimadores. Foi ativada uma flag para lidar com classes desbalanceadas;
- AdaBoost com 100 estimadores e o algoritmo SAMME.

- Os estimadores base usados são árvores de decisão com profundidade máxima de 5. Essas árvores de decisão possuem uma flag ativada para lidar com classes desbalanceadas.
- Regressão Logística com um máximo de 5000 iterações e uma flag para lidar com classes desbalanceadas.
- Bagging com 20 estimadores.
 - Os estimadores base usados são árvores de decisão com profundidade máxima de 15. Essas árvores de decisão possuem uma flag ativada para lidar com classes desbalanceadas.
- SVC (Support Vector Classification) Linear com uma flag ativada para lidar com classes desbalanceadas.
- Ensemble por votação.
 - Foram usados como estimadores base os modelos de MLP, Floresta Aleatória e Bagging com os mesmo hiperparâmetros citados anteriormente, com exceção do número máximo de repetições do MLP, o qual diminuimos para 15.
- SGD (Stochastic Gradient Descent).
- KNN (K Nearest Neighbors) com o valor de K igual a 20.
- Bernoulli Naive Bayes.
- LDA (Linear Discriminant Analysis).

Resultados Obtidos:

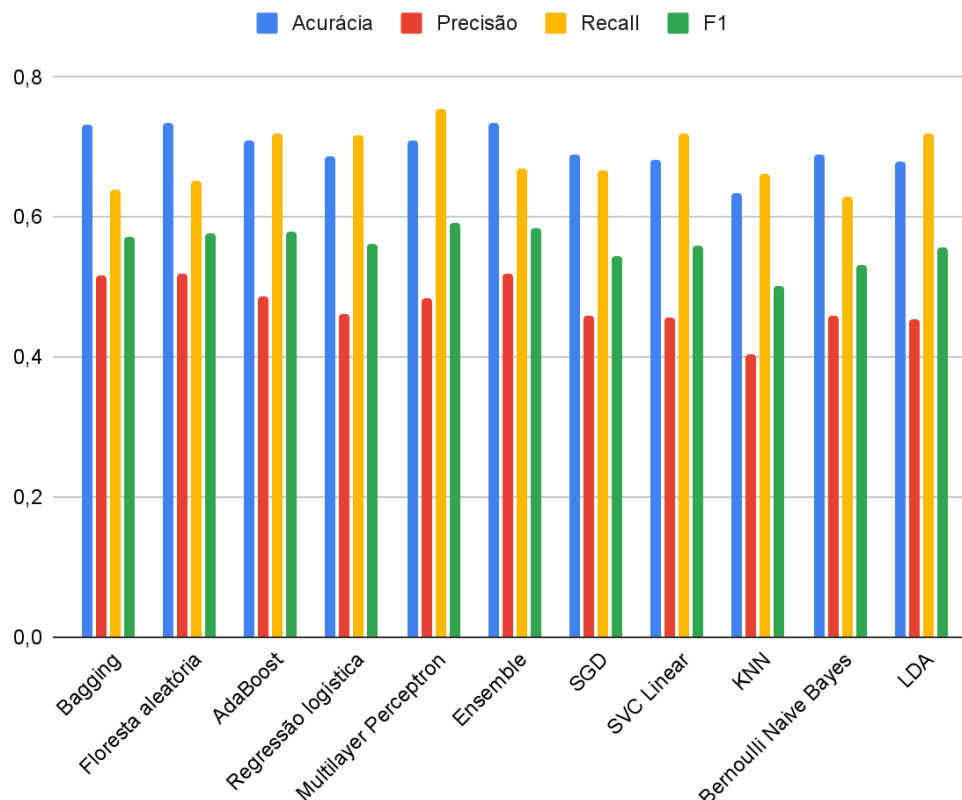


Figura 9: Resultados do Spot-Checking usando os dados pré-processados.

Podemos ver na Figura 9 o resultado dessa etapa de Spot-Checking. Podemos notar que o melhor resultado em termos de recall e da métrica F1 foi o do MLP. Os algoritmos de Adaboost e de Regressão Linear tiveram bons resultados com relação a essas métricas em comparação com os outros. Os melhores algoritmos com relação à acurácia foram o da Floresta Aleatória, Bagging e o da Regressão Linear.

Conclusões:

Através dessa etapa do trabalho, conseguimos compreender melhor a complexidade da tarefa de previsão do tempo. Mesmo com o problema escolhido sendo uma simplificação muito grande do problema de previsão, não conseguimos obter um modelo com um bom desempenho nessa primeira etapa do trabalho.

Por fim, conseguimos estimar modelos promissores para dar um foco maior nas próximas etapas, como o MLP, a Regressão Linear e o Adaboost, que tiveram um bom desempenho nas métricas escolhidas.

5. Referências

1. <https://portal.inmet.gov.br/dadoshistoricos>