Integração de uma solução de propensão de instalação de aplicativos em uma infraestrutura de Big Data organizacional

Lucas Ranieri[1], **Raissa P. P. M. Souza**[2], **Fabrício A. Silva**[1]

[1]Núcleo de Estudos em Sistemas Pervasivos e Distribuídos
Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa (UFV) – Florestal – MG – Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brazil

{lucas.ranieri,fabricio.asilva}@ufv.br, raissa.papini@dcc.ufmg.br

Abstract. *The process of integrating a solution into an existing cloud infrastructure requires some important steps to achieve functional and efficient synergy. There are necessary adaptations that permeate at different levels, ranging from changes in implementation details of the initial solution to integration into the infrastructure itself. When there is a characterization of Big Data, the use of cloud services ends up being a strong facilitator of aggregation of the system of interest. This is because both service demand issues and quality issues are addressed, which ends up facilitating the process of creating a cohesive and effective infrastructure. In this paper, a mobile application installation recommendation solution was adapted to consider segments, and deployed in an organizational Big Data environment of a data intelligence company. As a result, it was possible to apply the solution to more than 1 million users and verify the probability of installing each segment for each user.*

Resumo. *O processo de integração de uma solução em uma infraestrutura de nuvem já existente requer alguns passos importantes para se obter uma sinergia funcional e eficiente. Existem adaptações necessárias que permeiam em diversos níveis, abrangendo desde mudanças em detalhes de implementação da solução inicial até a integração na infraestrutura em si. Quando existe uma caracterização de Big Data, a utilização de serviços em nuvem acaba sendo um forte facilitador de agregação do sistema de interesse. Isso corre pois que são supridas tanto questões de demanda do serviço em si quanto questões de qualidade, o que acaba facilitando o processo da criação de uma infraestrutura coesa e efetiva. Neste trabalho, uma solução de recomendação de instalação de aplicativos móveis foi adaptada para considerar segmentos, e implantada em um ambiente de Big Data organizacional de uma empresa da área de inteligência de dados. Como resultado, foi possível aplicar a solução para mais de 1 milhão de usuários e verificar qual a probabilidade de instalação de cada segmento para cada usuário.*

1. Introdução

Atualmente o mercado de aplicativos de *smartphones* tem tomado uma grande proporção, como proposto em [mob 2022], o que gera um cenário de grande quantidade de dados disponíveis, que acabam gerando uma dificuldade no levantamento de tendências benéficas

para cliente e fornecedor. O principal motivo é que, diante de tantas informações, fica difícil separar o ruído dos indicadores relevantes. Dentre a grande quantidade de dados existente nesse cenário de *big data*, existe o surgimento e a descontinuação de vários aplicativos de diversos segmentos, levando a uma grande quantidade de dados de instalação e desinstalação. As instalações desses aplicativos por usuários podem representar uma boa base para um indicador de interesses com grande capacidade para geração de decisões de negócios.

A partir da premissa descrita, fica evidente a necessidade de levantamentos de bons sinalizadores de interesses dos usuários. O entendimento da propensão à instalação de aplicativos de determinados segmentos nasce, portanto, do desafio de gerar estratégias de divulgação e negócios direcionadas a usuários interessados em tipos específicos de serviços. É possível, então, utilizar a grande quantidade de dados disponíveis pelos usuários para gerar bons acordos.

O problema de identificar a propensão de instalação de aplicativos de *smartphones* já foi tratado em [Souza et al. 2021]. Nesse trabalho, foi proposta uma solução que utiliza LDA (*Latent Dirichlet Allocation*) para fazer a previsão de possível instalação de aplicativos por usuários. Essa previsão pode ser utilizada para que ações de divulgação e engajamento sejam feitas por parte das empresas. Porém, esse trabalho original não considera a escalabilidade em um cenário real de *big data*, e identifica a propensão de aplicativos específicos ao invés de segmentos, o que muitas vezes não é de tanto interesse para as empresas.

O fundamento do trabalho desenvolvido é baseado na produtização de um sistema de previsão de instalação de segmentos de aplicativos. O sistema base utilizado foi idealizado por [Souza et al. 2021]. Assim, adaptando o sistema original e com base em aplicações de diversas frentes presentes em *smartphones*, será possível estimar uma propensão de instalação de aplicativos que compõem tais segmentos, gerando um indicador de interesses dos usuários que possibilitará geração de negócios mais direcionados. O contexto por detrás das fontes de dados e infraestrutura utilizada é uma empresa parceira, que possui uma estrutura B2B (*Business to Business*), ou seja, seus artefatos são utilizados por outras empresas que possuem uma extensa base de clientes, que seriam os chamados "usuários com aplicativos", ou usuários finais. Assim, utilizando essas bases de dados, a empresa cria soluções de larga escala com infraestrutura de nuvem.

A contribuição do trabalho é criar uma solução escalável em uma arquitetura de nuvem, necessitando inicialmente de ajustes de código e da integração da solução com a infraestrutura de uma empresa parceira. O código base utilizado é uma *engine* de recomendação de instalação de aplicativos móveis, que foi construída com base em dados da mesma empresa cujo trabalho empreendido se deu. Somado a isso, a integração com uma estrutura de nuvem utilizada no mercado levantará alguns problemas que podem ser comuns a contextos parecidos. Um desses problemas é a própria questão da escalabilidade da solução, visto que cada cliente da empresa parceira possui uma enorme base de usuários com possivelmente diversas aplicações.

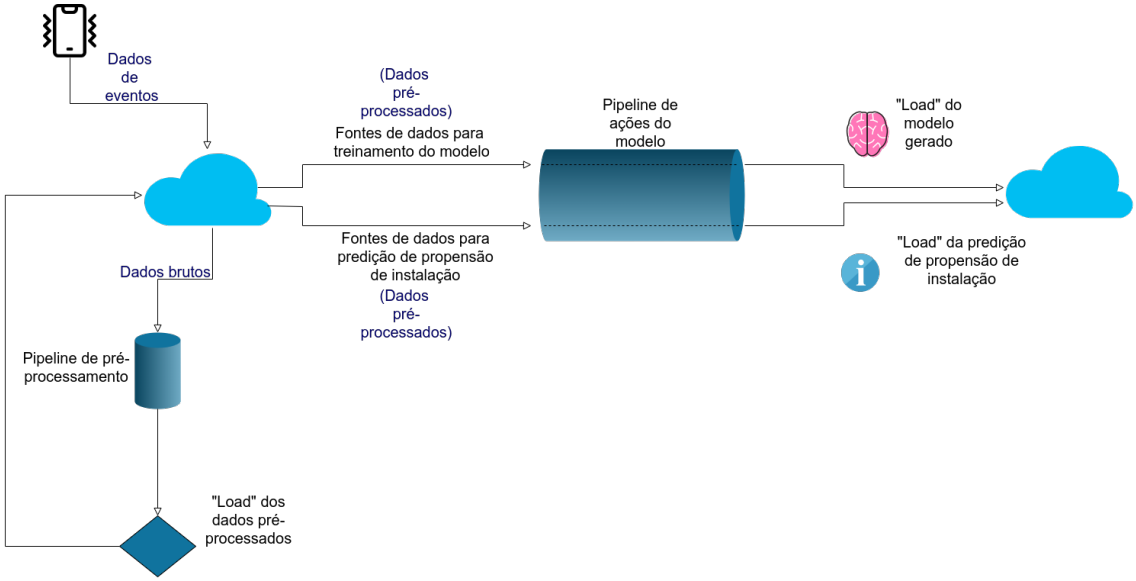


Figura 1. Visão geral dos componentes do sistema.

2. Solução

2.1. Visão Geral

Antes de adentrar no funcionamento interno da solução de propensão de instalação, é importante levantar como são os dados que serão utilizados. Os dados iniciais, gerados pelos processos internos da empresa parceira, possuem um campo de identificação do usuário final e um campo de listagem de identificadores dos segmentos nos quais as aplicações instaladas pelo usuário em questão fazem parte. Vale salientar que a listagem de identificadores de segmentos já é abstraída nos processos internos no qual são gerados os dados brutos utilizados como fonte para o pré-processamento de dados, como ilustrado na figura 1. Além disso, é utilizada a renda estimada do usuário com base em um enriquecimento de dados feito com fontes do IBGE.

A forma de operação básica para o sistema proposto de propensão de instalação de segmentos, que adapta a solução de recomendação de aplicações proposto por [Souza et al. 2021], pode ser entendida como constituinte de partes separadas, mas integradas. Essas partes são: treinamento, execução e retreinamento.

O treinamento opera com o intuito de gerar o modelo que será utilizado posteriormente para a execução, sendo ele representado na parte superior do pipeline de ações do modelo da figura 1. É importante destacar que o sistema precisa dos dados em um formato específico de entrada, assim, tanto para o treinamento quanto para a execução, os dados utilizados deverão ser pré-processados anteriormente de forma a atender tal requisito.

A execução lida, basicamente, com receber os dados que são gerados a partir do *pipeline* de dados interno da empresa parceira, modificar e passar esses dados alterados como entrada para o modelo e gerar as propensões de instalação, como apresentado na parte inferior do pipeline de ações do modelo da figura 1. É importante destacar que os dados de entrada contemplam cada usuário dos clientes, ou seja, esse *dataset* de entrada pode possuir um tamanho muito grande dependendo de para qual cliente a execução está sendo feita. Cada elemento desse *dataset* será um usuário do aplicativo que pertence a um

cliente específico da empresa parceira e que possui uma quantidade possivelmente grande de aplicativos instalados.

O retreinamento é uma operação mais simples, que consiste em atualizar o modelo de um determinado cliente para que as propensões geradas reflitam melhor a base de usuários em suas características mais atuais.

Uma importante característica associada tanto à execução quanto ao retreinamento é a regularidade do agendamento dessas tarefas, visto que a execução pode ocorrer com uma regularidade "média" e o retreinamento podendo ser agendado em uma frequência ainda menor. Isso pode ser proporcionado visto que as características do corpo de usuários de um cliente da empresa cujo trabalho se deu não possuem mudanças significativas em pouco intervalo de tempo.

2.2. Adaptações necessárias

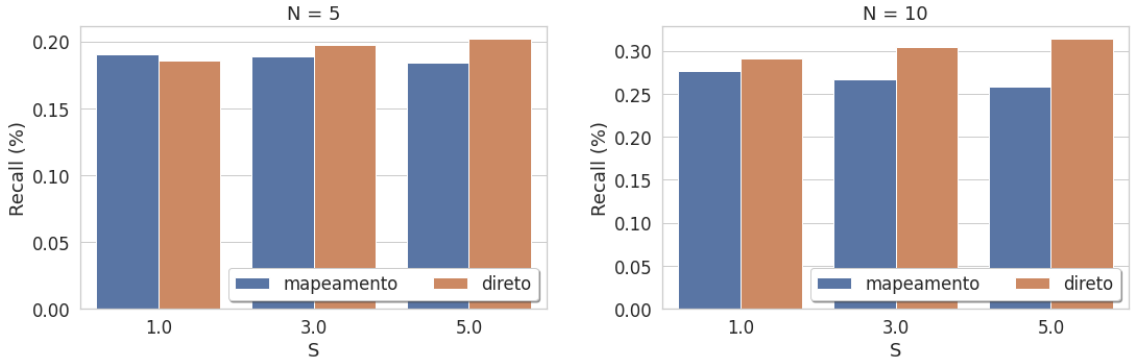
Um importante aspecto do trabalho conjunto com a empresa parceira é a questão da adaptação do código base. Nessa solução inicial, o modelo LDA (*Latent Dirichlet Allocation*) utilizado, recebe um *dataset* onde cada entrada representa um usuário final de um cliente da empresa parceira. Nesses dados, existe uma *feature* para identificar o usuário e uma outra com uma lista contendo seus aplicativos. A partir dessa entrada, o modelo busca, então, considerar cada aplicativo dos usuários como um termo e, a partir deles, são gerados tópicos no processo de treinamento. Cada tópico é uma estrutura que agrupa termos com mesmas "afinidades", ou seja, que são vistos no mesmo contexto. No caso dos aplicativos dos usuários, cada tópico iria agrupar apps instalados em conjunto ou com mesmo propósito (e.g., tópicos de aplicativos de vendas e de esportes). A partir do modelo já estabelecido, é possível então gerar o resultado final, que é a predição de propensão de instalação.

A primeira adaptação necessária foi na própria etapa de treinamento, e consequentemente retreinamento, do modelo gerado para cada cliente. O motivo essencial para tal adaptação se dá principalmente pelo fato do código base trabalhar com recomendações de aplicativos, e não segmentos nos quais esses aplicativos são caracterizados. Considerando a solução inicial, existe a possibilidade de fazer o mapeamento dessas aplicações para seus segmentos após a recomendação. Assim surgem duas possibilidades a serem avaliadas: gerar o modelo inicialmente baseado em aplicações e posteriormente fazer o mapeamento para seus segmentos; ou gerar, diretamente, o modelo baseado nos segmentos que o cliente possui instalado.

A diferenciação no método de treinamento está no nível de granularidade, visto que existe uma grande quantidade de aplicativos no mercado, mas pertencentes a um conjunto significativamente menor de segmentos. Assim, foi necessário checar o impacto positivo de um nível de magnitude maior ou menor. É importante salientar que o teste foi realizado ainda utilizando a solução base e foram utilizados cerca de 15 mil usuários finais de um cliente aleatório da empresa parceira, enriquecendo dados relativos à renda estimada do usuário, para ajudar no processo de treinamento e execução do modelo.

A partir de uma análise dos resultados do teste, foi checado que treinar diretamente com os segmentos manteve um resultado melhor, e, consequentemente, esse método foi escolhido como forma de treinamento. Essa conclusão pode ser observada nas figuras 2 e 3, onde a cor da barra indica o método de treinamento utilizado, o valor de N indica quan-

tos segmentos foram recomendados, o S indica quantos dos últimos aplicativos instalados pelos usuários serão utilizados para teste do modelo. É possível constatar, portanto, que treinar diretamente com os segmentos apresenta melhores resultados absolutos tanto de precisão quanto de revocação, onde o maior valor de revocação, treinando com segmentos, é 31,4% e fazendo o mapeamento o valor é 27,6%, como indicado pela figura 2. Já a maior precisão, treinando com segmentos, o resultado é 21,2% e fazendo o mapeamento o resultado é 18,4%, como indicado pela figura 3. Vale destacar que o cálculo da revocação leva em consideração todos os segmentos que foram recomendados, onde o N passa a ser um parâmetro importante. No caso da precisão, são considerados apenas uma quantidade de segmentos recomendados referente ao número de segmentos utilizados para teste. Assim, a revocação é uma métrica favorecida, visto que todos segmentos recomendados são importantes.

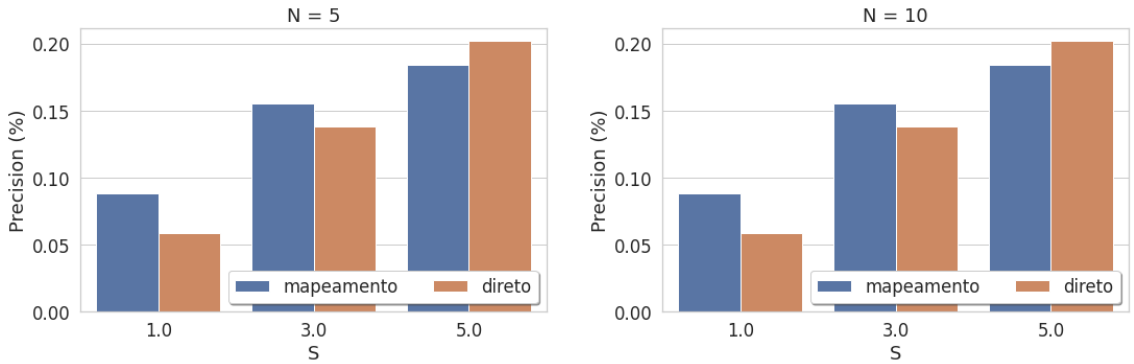


(a) N = 5 **(b)** N = 10

Figura 2. Valores de revocação na comparação do treinamento direto com segmento com o treinamento utilizando aplicações e posterior mapeamento.

O impacto principal da adaptação que abrangeu a forma de treinamento do modelo foi na etapa anterior à geração do modelo em si, o pré-processamento. O motivo principal desse ocorrido foi devido à necessidade de alteração na formatação dos novos dados de entrada.

Algumas importantes características que foram adicionadas à parte de execução



(a) N = 5 **(b)** N = 10

Figura 3. Valores de precisão na comparação do treinamento direto com segmento com o treinamento utilizando aplicações e posterior mapeamento.

do modelo são os formatos de saída. No código original, o formato de saída era relativamente simples, sendo constituído pelos tipos de dados: listas e objetos. Esse formato foi adaptado para retornar, agora, tanto as propensões de instalação de segmentos quanto um *score* que o próprio LDA associa aos segmentos que são gerados pelo modelo. Esse valor criado simboliza a representatividade desse segmento dentro de um determinado tópico, que pode ser entendido como um *cluster* criado para representar características distintivas. Vale ressaltar que todos os segmentos estão presentes em todos os tópicos, variando somente o *score* de acordo com o tópico.

3. Integração com infraestrutura da empresa

3.1. Pré-Processamento

A empresa parceira, no qual o trabalho foi realizado, possui em sua infraestrutura alguns *frameworks* internos para o processamento de dados. Dentre eles, existe um *framework* para consolidação de dados que possui seu funcionamento baseado em *Spark* para, principalmente, permitir processamento paralelo em *cluster*, como proposto originalmente por [Zaharia et al. 2010]. O *Spark*, como colocado por [Salloum et al. 2016], atualmente pertence à Fundação Apache, e atua como a *engine* de *big data analytics* e permite, dentre vários outros recursos, a distribuição automática de dados em um *cluster* e paralelização das operações requisitadas.

Para utilização das vantagens da computação distribuída, o *Spark* é utilizado através de sua interface para *Python*, denominada como *Pyspark*, que nada mais é que um pacote construído para se trabalhar com o core do *Spark* juntamente com suas bibliotecas constituintes, como proposto por [Salloum et al. 2016]. A principal vantagem da utilização de *Python* no contexto da empresa é sua facilidade de leitura e manipulação de dados, considerando principalmente o fato de ser uma linguagem de tipagem dinâmica com sintaxe bastante intuitiva.

A facilidade de manipulação de dados do *Python* é acompanhada de potentes formas de visualização de dados, como pacotes de gráficos e tabelas, o que permite grande vantagem para uma empresa que está associada à área de inteligência de dados. No contexto da integração da solução, o processamento paralelo, alavancado pela utilização do *Pyspark*, foi utilizado para a realização do pré-processamento dos dados. Como proposto anteriormente, o pré-processamento é uma etapa de grande importância, visto que essa permite a formatação dos dados presente na nuvem da empresa em um formato ideal para o funcionamento da solução de propensão de instalação de segmentos.

Os processos que compõe o pré-processamento podem ser divididos em etapas que englobam todas transformações e limpeza de dados necessárias para o posterior treinamento e utilização do modelo LDA. Essas etapas são: selecionamento de *features*, filtragem de usuários e transformações de dados. As duas primeiras etapas acarretam uma redução de dimensionamento e tamanho do *dataset*, que são necessárias para uma boa qualidade do modelo. Já a última etapa está associada a fazer a transformação dos dados de segmentos dos usuários e enriquecimento desses dados a renda aproximada. A transformação nos dados do segmento é necessária devido ao fato de que os dados brutos, internos da empresa parceira, apresentam apenas os identificadores dos segmentos, sendo necessário fazer consulta em um banco de dados relacional para transformar esses identificadores no nome dos segmentos.

3.2. Armazenamento

Para o armazenamento de seus diversos conjuntos de dados semi-estruturados, a empresa utiliza serviços de armazenamento em nuvem, como o S3[1] da AWS e o *Blob Storage*[2] da *Microsoft Azure*. O *Blob Storage* permite a utilização de sua estrutura para criação de um *Data Lake* para armazenamento dos dados principais que são gerados pelos processos internos. A organização no *Blob Storage* para o pré-processamento e para a predição de propensão de instalação são em termos dos clientes da empresa parceira, sendo que para cada cliente, existem dados pré-processados e, após a execução do modelo, processados.

Um importante ponto a se destacar são os formatos de arquivos que são armazenados nos serviços de armazenamento mencionados até então. Inicialmente, o projeto base, sendo uma *engine* de recomendação, se dispunha do formato CSV como seu principal meio para receber dados de entrada. Entretanto, como o sistema idealizado está presente num contexto de *Big Data*, a utilização de um formato mais eficiente se faz necessária. Com isso entra o *Parquet*[3], que é um formato *open-source* baseado em colunas utilizado para armazenamento, leitura e escrita eficiente. Como apresentado por [de Oliveira et al. 2021], o *Parquet* apresenta, comparado a CSV que é baseado em linhas ao invés de coluna, resultados melhores e mais eficientes em consultas, principalmente considerando um cenário de grande quantidade de dados.

3.3. Big Data

Quando se trata de um contexto onde a quantidade de dados é um fator importante a se considerar, entra em questão o problema de "como facilitar o processamento de uma grande quantidade de dados". Para lidar com problemas como esses existem diversos serviços disponíveis em infraestruturas de nuvem, como AWS, GCP e *Azure*. Tais plataformas facilitam o trabalho de diversas empresas, onde não é necessário arcar com detalhes e cuidados necessários para manter localmente uma grande estrutura de armazenamento e processamento de dados.

No contexto de processamento, as diversas plataformas citadas oferecem serviços de processamento baseado em *clusters*, onde passa a não ser necessário ter uma estrutura própria com vários núcleos de processamento, bastando apenas "instanciar" tais núcleos na plataforma desejada. O serviço utilizado, em questão, para realizar o pré-processamento baseado em *Spark*, foi o *HDinsight*[4], que permite utilização de *cluster* para processamento de grandes volumes de dados.

Para os arranjos relacionados ao próprio modelo utilizado, baseado em LDA, pode ser usado tanto *cluster* quanto as próprias máquinas virtuais disponibilizadas por serviços presentes nas plataformas mencionadas até então. Isso ocorre devido ao fato de que o código escrito para o modelo, tanto o original quanto o adaptado, não está em *Pyspark*, portanto, a utilização de *cluster* não afetaria tanto o desempenho do processamento. O motivo da não utilização de *Pyspark* no código adaptado é principalmente a dificuldade de manter o padrão de estrutura do código da empresa parceira com tal abordagem.

[1] docs.aws.amazon.com/

[2] docs.microsoft.com/pt-br/azure/storage/blobs/storage-blobs-introduction

[3] parquet.apache.org/

[4] docs.microsoft.com/pt-br/azure/hdinsight/hdinsight-overview

3.4. Integração

Como já discorrido anteriormente, a empresa parceira dispõe de *frameworks* internos para lidarem com o processamento de grandes volumes de dados. Dentre esses sistemas, já foi mencionado que existe um específico para lidar com processamento de dados gerados internamente com a utilização de *Spark*, que foi utilizado para abrigar a parte da solução de propensão de instalação relativa ao pré-processamento. Esse componente atuará sobre os dados internos para gerar um formato compatível com o que será "entregue" tanto para o modelo gerado realizar as predições de propensões de instalações como para a realização do próprio treinamento do modelo aspirado.

Para realização de tarefas relacionadas ao modelo que será gerado em si, existe um outro *framework* interno que é responsável por lidar com quaisquer questões associadas a treinamento e execução de modelos. Uma diferença entre esses dois sistemas se dá pelo fato de que não é necessário a utilização de *Spark* para questões relacionadas a modelos. Assim, essa solução interna permite a utilização de outras bibliotecas e módulos específicos. Isso se apresentou como uma vantagem, visto que o algoritmo inicial, ainda como uma *engine* de recomendação, foi escrito utilizando bibliotecas como *Gensim*[5], para utilização do próprio algoritmo de aprendizado de máquina não supervisionado LDA; e o próprio Pandas, para lidar com os *data frames* iniciais, intermediários e finais que são gerados pelo algoritmo. As atribuições que esse sistema em específico possui são em termos de treinamento do modelo, utilizando os dados resultantes da etapa de pré-processamento, e da execução do modelo para cada usuário considerado no pré-processamento.

É importante destacar que, da mesma forma em que acontece para os dados relativos ao pré-processamento e processamento final, existe um modelo para cada cliente da empresa parceira, que proporciona predições mais acuradas, visto que cada cliente possui características que o distingue dos demais. Esses modelos são salvos, igualmente, no *Blob Storage* da *Microsoft Azure*, em uma organização que se dá por clientes.

Para informar ou parametrizar a forma no qual o pré-processamento irá ser feito, existe um arquivo de configuração no formato JSON que é utilizado como base no *framework* interno utilizado pela empresa. Nesse arquivo, são indicadas informações de parâmetros, fontes de dados e carregamento dos dados. Os parâmetros indicam informações básicas relacionadas ao *job Spark*, que consiste nas tarefas realizadas utilizando o *Pyspark*. Dentre essas informações existe qual nuvem será utilizada, dados de cliente e data dos dados que serão utilizados para realizar o pré-processamento. As fontes de dados, como o nome sugere, informam quais dados serão utilizados como base para o pré-processamento. Além disso são informadas a configuração e disposição no qual esses dados se apresentam na nuvem. Por fim, as configurações de carregamento indicam onde e como os dados finais, gerados pelo *job* criado, serão armazenados também na nuvem indicada.

Da mesma forma que ocorre no *framework* anterior, existem também arquivos de configuração responsáveis por indicar informações de parâmetros, fontes de dados e carregamento dos dados para aquele relacionado ao treinamento e execução. Porém, somado a essas informações, existe a possibilidade de indicar modelos como fonte de dados e como informação a ser carregada de volta para a nuvem. Em relação a forma no qual

[5] radimrehurek.com/gensim/index.html

os modelos serão salvos, foi utilizado o *Joblib*, que é um conjunto de utilitários de *pipeline* de dados, permitindo, dentre outras coisas, persistir objetos *Python* com eficiência. No caso do treinamento, existirá um arquivo de configuração específico, que indicará os parâmetros, fontes de dados, e o carregamento do modelo. Já no caso da execução, o arquivo de configuração será diferente, nele o modelo será indicado como uma fonte de dados que estará estruturada na nuvem especificada nos dados de parâmetro.

No contexto da empresa parceira, a questão de retreinamento é facilitada devido ao fato de que os dados de interesse, gerados pelos *pipelines* internos da empresa, são históricos. Ou seja, além das informações novas que foram acrescidas a partir de suas fontes de dados, são mantidos os dados antigos no mesmo conjunto de arquivos no formato *Parquet*. Além disso, os dados utilizados são organizados por data numa granularidade diária, o que facilita ainda mais. A partir disso, é possível, então, para realização do treinamento do modelo, apenas treinar novamente atualizando a fonte de dados para uma data mais recente.

3.5. Caso de Teste

Para melhor compreensão sobre o funcionamento do sistema discutido até então, foi criado um caso de teste utilizando um cliente aleatório da empresa parceira. É importante citar que tal teste foi executado na mesma infraestrutura especificada anteriormente.

Para cliente escolhido, foi utilizada uma base de cerca de 6.5 milhões usuários finais. Entretanto, foram retirados usuários que não possuíam informações dos segmentos instalados, resultando em cerca de 1.05 milhões de usuários, que nos geram uma base útil de cerca de 16% da original. O motivo da falta de informação do restante da base podem ser de várias naturezas. Por exemplo, se um usuário deixa de ser ”assinante”do cliente, seus dados são limpos no sistema, mas ele ainda tem seu *id* persistido. Por esse e outros motivos podemos ter dados que não são diretamente úteis para treinamento e execução.

Como pode ser observado na figura 4, temos alguns segmentos que possuem mais incidência de propensão de instalação. Dentre eles podem ser citados o segmento de *E-commerce*, com cerca de 430 mil recomendações, Serviços Financeiros, com cerca de 410 mil recomendações e Bancos Digitais, com cerca de 386 mil recomendações.

A partir da figura 5, podemos ter uma visão mais profunda do sistema, pois a partir dela é possível observar a variação do *score* que cada segmento possui em possíveis tópicos gerados pelo sistema. Isso ocorre devido ao fato de que, como já mencionado, todos os tópicos gerados pelo LDA contemplam todos segmentos, mas atribuindo valores distintos a cada um deles.

4. Conclusão

O trabalho apresentado permitiu observar as nuances do que se propõe uma implementação de um sistema em uma infraestrutura de *Big Data*, permitindo adentrar em diversas tecnologias distintas que cobrem diferentes aspectos da solução, principalmente aqueles relacionados a serviços de nuvem e processamento paralelo. Inicialmente, foi possível se utilizar do Pyspark para realização de um pré-processamento bastante otimizado a partir dos dados internos da empresa parceira. Se utilizando dos dados do pré-processamento, que são armazenados no *Blob Storage*, são feitos o treinamento e a

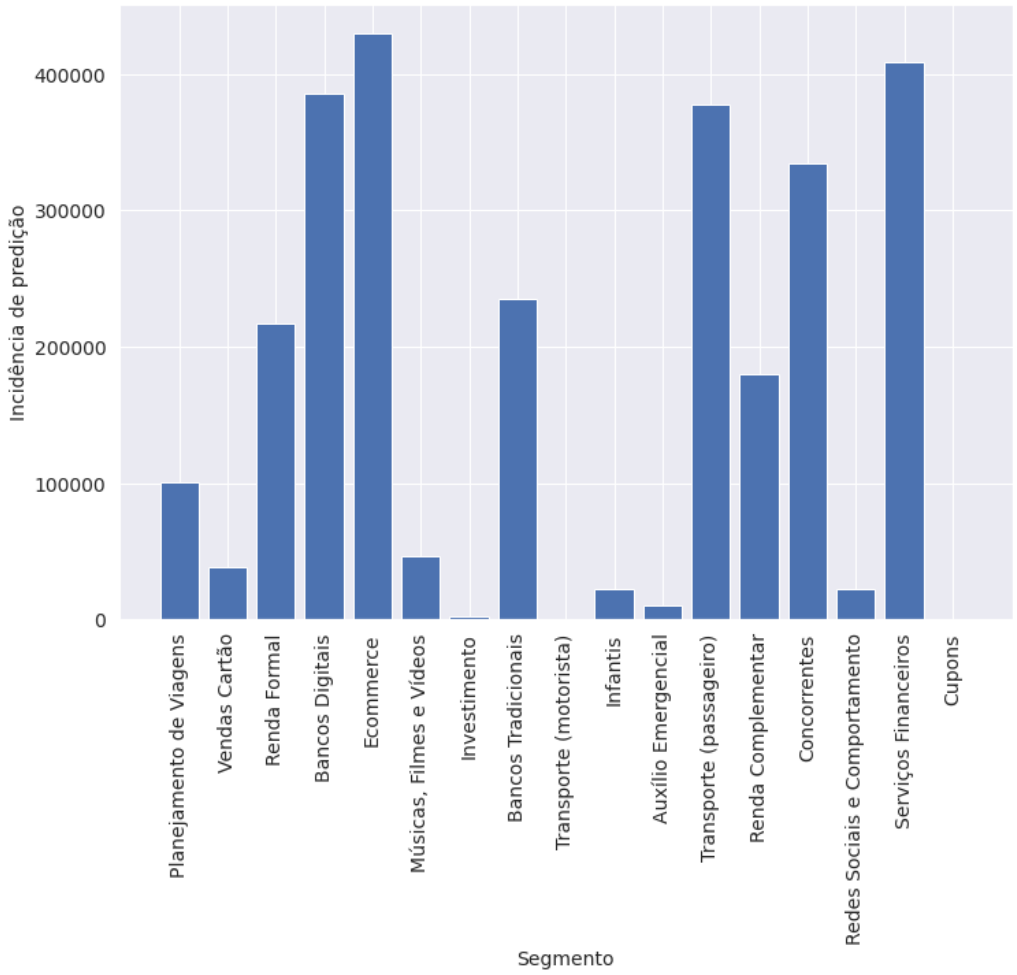


Figura 4. Número de incidência de propensão de instalação para os segmentos do cliente escolhido.

posterior execução do modelo. A partir disso, todos esses resultados, tanto o modelo gerado quanto a predição realizada são adicionados também no *Blob Storage*. Assim, foi possível gerar resultados de execução, tanto um modelo gerado quanto as propensões de instalação de segmentos, com a utilização de cerca de 1.05 milhões de usuários de um cliente aleatório da empresa parceira.

O projeto apresentado demonstra relevância pelo seu aspecto de materialização de um trabalho científico de base dentro da praticidade presente no ambiente das organizações no mercado de trabalho. A partir dessa abordagem, foi possível notar os desafios que se apresentaram através de adaptações necessárias para conciliação da solução em uma infraestrutura existente e a complexidade de lidar com o problema de escalabilidade devido ao cenário de *big data* que a empresa parceira proporcionou. O trabalho cumpre, então, seu objetivo de integração de uma solução escalável se utilizando de diversas ferramentas que contornam os problemas que se apresentaram.

Em termos de melhorias possíveis do trabalho realizado, a utilização de processamento paralelo para geração, e, principalmente, execução do modelo se apresenta como uma possibilidade de otimização. Para isso, seria possível utilizar das vantagens já apre-

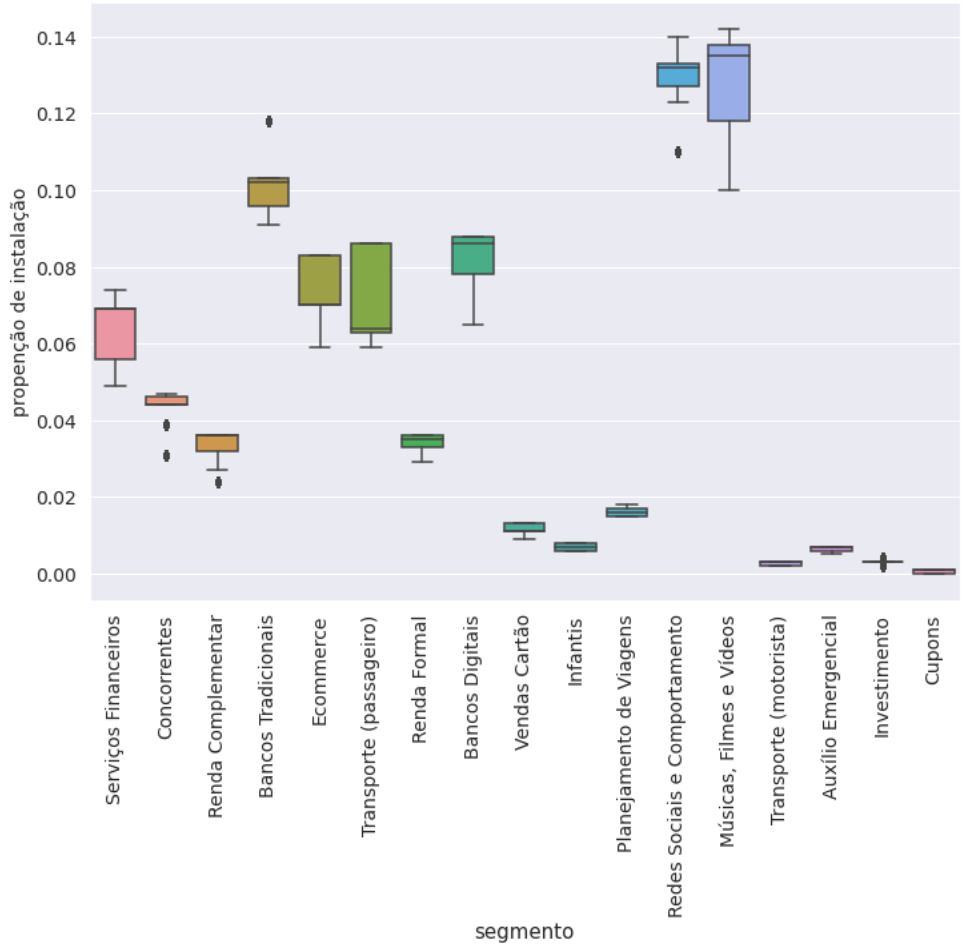


Figura 5. Número de incidência de propensão de instalação para os segmentos do cliente escolhido.

sentadas na utilização do *PySpark*, que possibilitou a execução bastante rápida do pré-processamento do modelo LDA.

Referências

(2022). The Mobile Economy. [Online; accessed 24. Jul. 2022].

de Oliveira, B. F. P., Valente, A. S. O., Victorino, M., Ribeiro, E., and Holanda, M. (2021). Análise da influência da modelagem e formato de dados no desempenho de data warehouse baseado em hadoop-hive. In *Anais do XXXVI Simpósio Brasileiro de Bancos de Dados*, pages 271–276. SBC.

Salloum, S., Dautov, R., Chen, X., Peng, P. X., and Huang, J. Z. (2016). Big data analytics on apache spark. *International Journal of Data Science and Analytics*, 1(3):145–164.

Souza, R. P. P. M., Coimbra, G. T. P., Figueiredo, L. J. A. S., Silva, F. A., and Silva, T. R. M. B. (2021). Mobile application recommendation based on demographic and device information. WebMedia '21, page 105–112, New York, NY, USA. Association for Computing Machinery.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*.