UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

LUIZ GUSTAVO WEBER THUMS

MODELO BASEADO EM APRENDIZADO DE MÁQUINA PARA RECOMENDAÇÃO DE VEÍCULOS DE FROTAS DE TRANSPORTE COLETIVO

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2021

LUIZ GUSTAVO WEBER THUMS

MODELO BASEADO EM APRENDIZADO DE MÁQUINA PARA RECOMENDAÇÃO DE VEÍCULOS DE FROTAS DE TRANSPORTE COLETIVO

MACHINE LEARNING-BASED MODEL FOR RECOMMENDING PUBLIC TRANSPORT FLEET VEHICLES

Trabalho de Conclusão de Curso apresentada como requisito para obtenção do título de Tecnólogo em Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Fabio Alexandre Spanhol
UTFPR-TD

TOLEDO
2021

LUIZ GUSTAVO WEBER THUMS

MODELO BASEADO EM APRENDIZADO DE MÁQUINA PARA RECOMENDAÇÃO DE VEÍCULOS DE FROTAS DE TRANSPORTE COLETIVO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnológo em Sistemas para Internet da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 01/dezembro/2021

_____

Fabio Alexandre Spanhol
Doutor
Universidade Tecnológica Federal do Paraná

_____

Jefferson Gustavo Martins
Doutor
Universidade Tecnológica Federal do Paraná

_____

Sidgley Camargo de Andrade
Doutor
Universidade Tecnológica Federal do Paraná

TOLEDO

2021

Dedico este trabalho a quem colaborou diretamente comigo: meu orientador, o Professor Fabio Alexandre Spanhol, sem o qual eu não teria concluído este projeto.

AGRADECIMENTOS

A minha família e a minha noiva por todo o apoio, carinho e por sempre acreditarem no meu potencial.

Ao meu orientador, Prof. Fabio Alexandre Spanhol, pela oportunidade, orientação e aprendizado durante esse trabalho. Aos demais docentes da UTFPR-Toledo pelas sugestões e colaborações durante minha formação acadêmica. A UTFPR-Toledo foi minha segunda casa durante o período da graduação.

Ao meu amigo, o professor Celso Costa, que contribuiu com muitos *insights* na condução da pesquisa, apoiando-me e ajudando a dirimir muitas dúvidas comuns de um iniciante nesta área.

Eu denomino meu campo de Gestão do Conhecimento, mas você não pode gerenciar conhecimento. Ninguém pode. O que pode fazer – o que a empresa pode fazer – é gerenciar o ambiente que otimize o conhecimento. (PRUSAK, Laurence, 1997).

RESUMO

THUMS, Luiz Gustavo Weber. Modelo Baseado em Aprendizado de Máquina para Recomendação de Veículos de Frotas de Transporte Coletivo. 2021. 50 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Toledo, 2021.

Atualmente o sistema de transporte coletivo é um dos principais meios de mobilidade urbana dentro das cidades. Entretanto, tal sistema em grandes centro urbanos apresenta diversos desafios, sobretudo a crescente superlotação dos veículos, impactando negativamente na comodidade e até na segurança dos usuários do sistema. O objetivo principal deste trabalho é propor um modelo de aprendizado de máquina baseado em dados históricos que ajude a otimizar o uso da frota de veículos de transporte coletivo urbano. Para validar o modelo foram conduzidos experimentos utilizando algoritmos baseados em séries temporais visando recomendar dinamicamente o melhor tipo de veículo a ser utilizado considerando efeitos sazonais e recorrentes. Verificou-se que o modelo conseguiu recomendar veículos melhor dimensionados a partir das demandas de volume do uso do veiculo.

Palavras-chave: Transporte coletivo urbano. Aprendizado de máquina. Sistema de recomendação.

ABSTRACT

THUMS, Luiz Gustavo Weber. Machine Learning Based Model for Recommending Collective Transportation Vehicles. 2021. 50 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Toledo, 2021.

The public transport system is one of the main problems of urban mobility within cities today. One of the biggest difficulties that is experienced in day-to-day life, both in large centers and in almost every city that has a public transportation system, is the overcrowding in the vehicles of this system. The objective of this work is to present a proposal for a Machine Learning model, which can be used to improve the dimensioning, so that vehicles compatible with demand are used. During this study, experiments will be carried out using Machine Learning algorithms, applied to time series, to recommend the best type of vehicle to be used during the operation.

Keywords: Collective transport. Machine learning. Recommendation system.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação
SVM	*Support Vector Machine*
ML	*Machine Learning*
PV	Positivo Verdadeiro
FP	Falso Positivo
FN	Falso Negativo
NV	Negativo Verdadeiro
SBE	Sistema de bilhetagem eletrônica

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta a contextualização da pesquisa, a motivação e os objetivos esperados. Ao término é apresentada a organização do restante do documento.

1.1 CONTEXTUALIZAÇÃO

Nos últimos 15 anos, com a melhoria de infraestrutura nos negócios, foi incrementada a capacidade de coletar dados na maioria das empresas. Atualmente, praticamente todos os aspectos de negócios encontram-se abertos para a coleta de dados e muitos setores beneficiam-se disso como, por exemplo, operações, manufatura, cadeia de suprimento, comportamento dos clientes, campanhas de marketing, etc. Esses setores têm angariado benefícios com a coleta de dados, pois, com o aumento da disponibilidade de dados surgiu o interesse em métodos para a extração de informações úteis e conhecimento a partir de tais dados, a denominada ciência de dados (PROVOST; FAWCETT, 2013). Neste contexto, aprendizado de máquina (ML–*Machine Learning*) é um dos subcampos da ciência da computação que evoluiu do estudo do reconhecimento de padrões e da grande área da inteligência artificial. Arthur Samuel, ainda em 1959, descreveu aprendizado de máquina como sendo o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados (SIMON, 2013). Essa habilidade, em conjunto com as ferramentas de coleta de dados, permite que sejam tomadas decisões não só considerando padrões e comportamentos atuais, mas também analisar dados passados permitindo uma previsão de movimentações futuras (RUSSELL; NORVIG; DAVIS, 2013).

1.2 OBJETIVOS

Diante do exposto na seção precedente evidencia-se a relevância do tema, dado o impacto deste no cotidiano urbano e a quantidade reduzida de pesquisas realizadas na área de transporte coletivo urbano. Assim, faz-se necessária a realização de estudos para aplicação de algoritmos de ML, para a indicação e otimização do uso da frota baseado na necessidade de um determinado tipo de veículo, para atender a demanda de passageiros e evitar superlotações .

1.2.1 Objetivo geral

A pesquisa tem o objetivo de gerar um modelo de indicação de veículos para atender as demandas da melhor maneira possível, e evitar problemas muito comum em várias cidades que são a lotação de veículos do transporte coletivo.

1.2.2 Objetivo especifico

(a) Realizar a extração de dados do sistema de bilhetagem eletrônica;
(b) Selecionar os modelos de aprendizado de máquina que melhor se aplicam na resolução do problemas;
(c) Gerar um modelo de previsão de tipo de veiculo;
(d) Avaliação do modelo gerado.

1.3 JUSTIFICATIVA

O trabalho é relevante, pois pode melhorar o nível de satisfação do usuário do transporte coletivo e, ao mesmo tempo, reduzir o custo operacional da empresa, visto que a indicação de um veículo menor, para uma linha com baixa demanda reduz o custo por quilômetro desta operação, da mesma forma a tarifa pode vir a sofrer um impacto haja visto a possível redução do custo por quilômetro. Além disso, existem poucos estudos utilizando-se também de algoritmos de ML aplicados a séries temporais voltados especificamente para a área de transporte coletivo. Em um trabalho correlato publicado por Xue, Sun e Chen (XUE; SUN; CHEN, 2015) o objetivo principal é fazer previsões precisas de ônibus de curto prazo frente a demanda de passageiros para ajudar a aumentar a eficiência da operação de ônibus e também minimizar o custo da operação, finalisticamente visando melhorar a qualidade e a confiabilidade do serviço. Ressalta-se, entretanto, que diferentemente do presente trabalho, esse estudo aplica-se somente a uma rota de ônibus.

1.4 ORGANIZAÇÃO DO TRABALHO

O restante deste trabalha está organizado da seguinte forma. No Capítulo 2 é apresentada a fundamentação teórica, revisando as principais técnicas adotadas para a realização deste trabalho. No Capítulo 3 é apresentada a metodologia adotada em cada uma das etapas. No Capítulo 4 é apresentado os resultados obtidos no decorrer deste trabalho. No Capítulo 5 são apresentadas as considerações finais.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta uma revisão das principais tecnologias utilizadas neste trabalho. Inicialmente é apresentada a mineração de dados. Na sequência, aprendizado de máquina e as duas técnicas de aprendizado mais utilizadas: supervisionado e não supervisionado. Em seguida são abordados os algoritmos utilizados para elaborar os modelos de aprendizado: *Support Vector Machine* (SVM), *Random Forest*, *Naive Bayes* e Regressão Logística. As métricas utilizadas para avaliar a eficência do modelo gerado também serão revisadas.

2.1 MINERAÇÃO DE DADOS

A mineração de dados é o processo de descobrir padrões em grandes conjuntos de dados que envolvem métodos na interseção de sistemas de aprendizado de máquina, estatística e banco de dados. A mineração de dados é a etapa de análise do processo "descoberta de conhecimento em bancos de dados-- KDD (*Knowledge Discovery in Databases*). Tais padrões podem ser vistos como uma espécie de resumo dos dados de entrada e podem ser usados em análises adicionais ou no aprendizado de máquina e na análise preditiva (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

2.2 APRENDIZADO DE MÁQUINA

A área de aprendizado de máquina é o estudo da capacidade de como um sistema computacional pode "aprender"tarefas específicas, como detecção de padrões, reconhecimento de caracteres, apoio a diagnóstico de doenças, etc. Essas e muitas outras aplicações evidenciam a utilidade do uso de aprendizado de máquina para a resolução de problemas diários além de apoiar os especialistas em tomadas de decisões, atraem pesquisadores das diversas áreas de conhecimento (MELLO; PONTI, 2018).

2.2.1 Aprendizado supervisionado

No aprendizado supervisionado, o algoritmo de ML recebe exemplos de entradas já rotulados e busca os melhores classificadores, para que seja possível prever novos rótulos, para dados não vistos com boa acurácia (MELLO; PONTI, 2018).

Através de um bom conjunto de treinamento é possível medir a precisão de um rótulo, por esse motivo é fornecido um conjunto de testes com exemplos distintos do conjunto utilizado no treinamento, uma hipótese se generaliza bem se prevê corretamente o valor do rótulo para novos exemplos (RUSSELL; NORVIG; DAVIS, 2013).

2.2.2 Aprendizado não supervisionado

Na aprendizagem não supervisionada, o algoritmo de ML procura padrões em dados não rotulados previamente, o modelo é construído após análise de semelhança entre os dados de entrada (MELLO; PONTI, 2018).

2.3 SUPPORT VECTOR MACHINE (SVM)

Uma Máquina de Vetores de Suporte (*Support Vector Machine* - SVM) reúne vantagens de modelos lineares e não lineares. Um algoritmo de SVM constrói um hiperplano ou um conjunto deles em um espaço de alta dimensão, que pode ser usado para regressão, classificação ou detecção de *outlier*. A priori obtém-se uma boa separação pelo hiperplano que tenta maximizar a distância entre as instâncias de cada classe mais próximas da separação. Geralmente quanto maior a margem, menor o erro de generalização do classificador (HASTIE; TIBSHIRANI; FRIEDMAN, 2001). Para ilustrar, na Figura 1 pode-se observar o algoritmo de SVM demonstrado de forma linear, e na Figura 2 o algoritmo é demonstrado de forma não linear

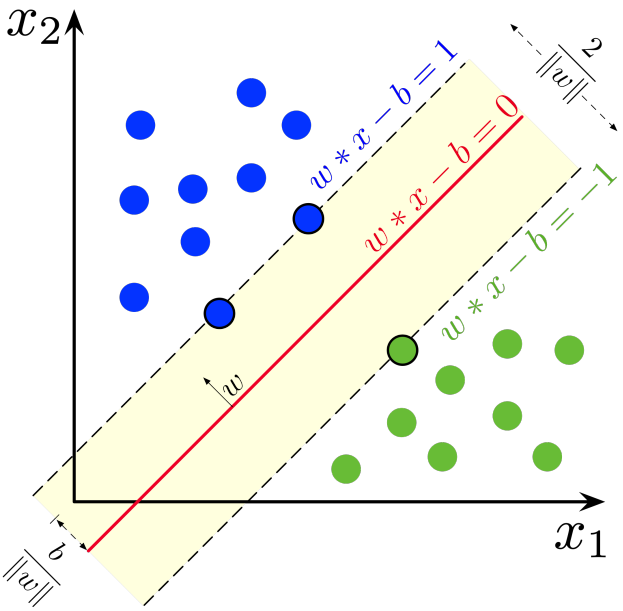
A seleção dos parâmetros de *kernel* do SVM são importantes para a obtenção de um bom desempenho. Recomenda-se a escolha manual dos parâmetros do *kernel*, baseado no conhecimento do conjunto de dados a ser avaliado, esse processo de selecionar parâmetros baseado no conhecimento prévio do conjunto de dados pode gerar uma acurácia inferior, para resolver esse problema da seleção manual a literatura propõe o uso de métodos automáticos, uma técnica muito utilizada quando não há o conhecimento prévio do conjunto de dados é utilizado a busca de grade (BONESSO, 2013).

A busca de grade é o método de busca exaustiva de um agrupamento do espaço de trabalho, onde a seleção de parâmetros do *kernel* RBF é formada pela seleção do parâmetro *gamma* γ e do parâmetro custo C, as coordenadas deste espaço são formadas por um par ordenado (γ, C). O objetivo da busca exaustiva é encontrar um espaço formado por (γ, C) pontos onde a acurácia seja a maior possível (BONESSO, 2013).

2.4 RANDOM FOREST

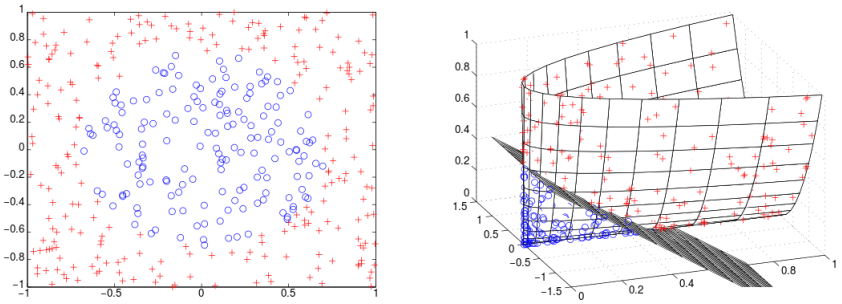
Random Forest é um método de aprendizado utilizado tanto para classificação quanto para regressão. Esse algoritmo cria múltiplas árvores de decisão onde cada nó da árvore representa um teste e cada aresta representa os possíveis resultados. Para que seja classificada uma instância deve-se apresentar o item para avaliação no primeiro nó da primeira árvore, o resultado determina qual será o próximo nó a ser utilizado e assim sucessivamente até chegar a uma folha, na qual o item será rotulado. Após o item passar por todas as árvores de decisão criadas e receber os rótulos, o algoritmo Random Forest analisa todos os rótulos encontrados nas folhas e seleciona o rótulo de maior ocorrência (FORSYTH, 2018). O funcionamento mencionado a cima é demonstrado na Figura 3

Figura 1 – Exemplo do algoritmo SVM linear



Fonte: Wikipedia (2020)

Figura 2 – Exemplo de algoritmo SVM não linear

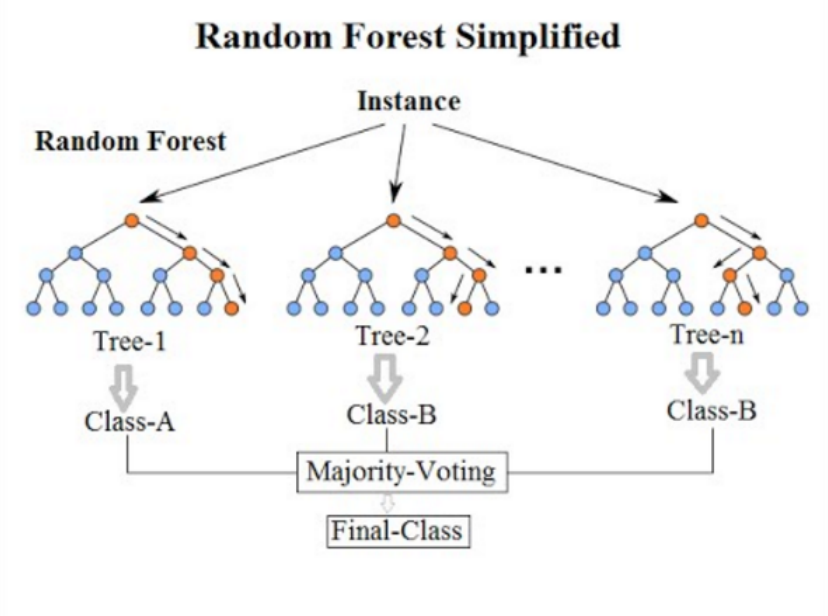


Fonte: Wikimedia (2017)

2.5 NAIVE BAYES

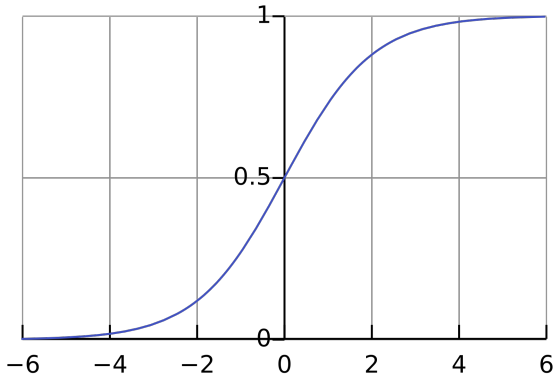
O algoritmos de classificação naive Bayes é um classificador probabilístico, baseado no teorema de bayes, este algoritmo é considerado simples devido utilizar a probabilidade de uma hipótese dado alguma evidência. É um classificador que possui um bom desempenho apesar de sua simplicidade, devido a boa capacidade de ser um algoritmo de aprendizagem incremental. Uma aplicação comum é em sistemas para classificação de mensagens de *spam* em serviços de email (PROVOST; FAWCETT, 2013).

Figura 3 – Exemplo do algoritmo de Random Forest



Fonte: Wikimedia (2020)

Figura 4 – Exemplo de uma regressão logística



Fonte: Wikimedia (2008)

2.6 REGRESSÃO LOGÍSTICA

Regressão logística é um método estatístico que utiliza um conjunto de observações para gerar um modelo capaz de predizer novos valores. Esse método utiliza uma função logística para modelar a partir de observações as relações entre uma variável (classe) e uma série de variáveis explicativas que podem ser numéricas ou discretas (CABRAL, 2013). A regressão logística, encontra o melhor modelo que descreve a relação entre variáveis, a variável de saída (variável dependente) e as variáveis independentes(preditoras ou explanatórias), onde as variáveis de saídas são binárias (0 ou 1) (HOSMER; LEMESHOW; STURDIVANT, 2013).Na Figura 4 podemos ver um exemplo de uma regressão logística conforme citado no texto acima

2.7 CURVA ROC

A curva ROC é um método para avaliar o desempenho de um classificador binário, e possui dois parâmetros:

- Taxa de verdadeiro positivo que é medido da seguite forma:

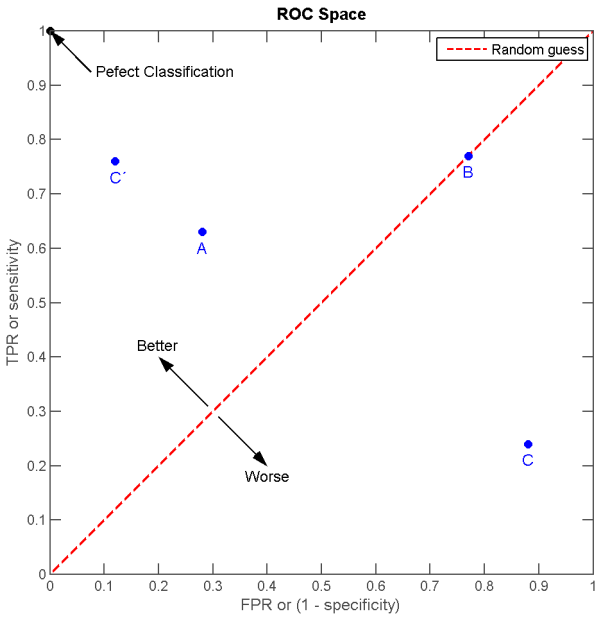
$$TVP = \frac{VP}{(VP + FN)}$$

- Taxa de falso positivo que é medido da sequinte forma:

$$TFP = \frac{FP}{(FP + VN)}$$

Através da análise de curva ROC é possível selecionar um modelo ideal e descartar modelos que não são tão bons (PROVOST; FAWCETT, 2013). Na Figura 5 temos um exemplo da curva ROC explicado acima.

Figura 5 – ROC



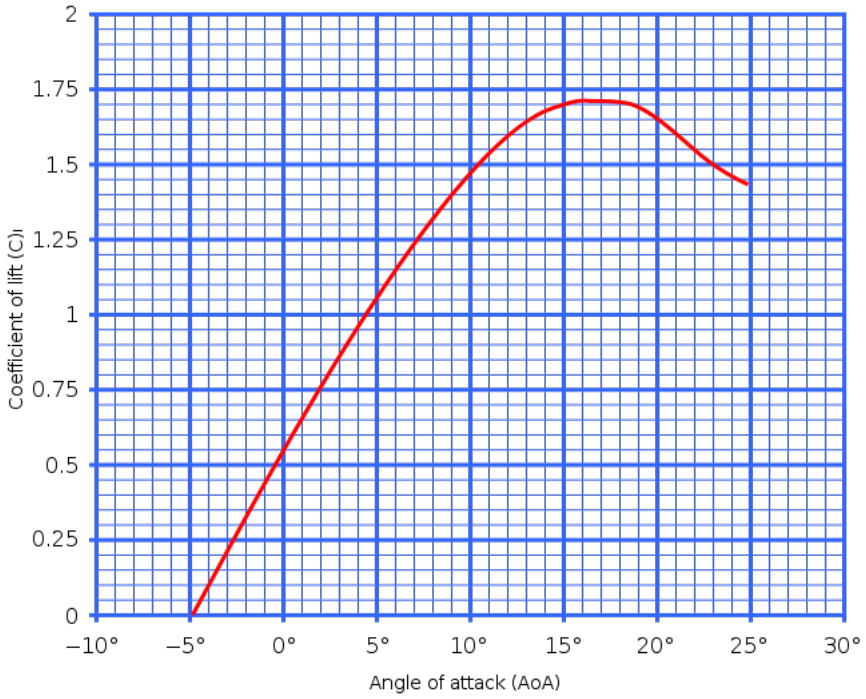
Fonte: Wikimedia (2009)

2.8 CURVA LIFT

Lift é uma medida de efetividade de um modelo preditivo calculada como sendo a taxa entre os resultados obtidos com e sem o modelo preditivo. Assim como ROC, o gráfico da curva de lift auxilia a avaliar visualmente um modelo. Sendo uma variação da curva ROC, já há algum tempo tem aplicações em áreas como marketing e econometria. Também conhecida

como curva de Lorenz ou curva de potência, a curva lift representa a proporção de x eventos detectados e função da proporção das seleções de indivíduos que tenham a pontuação maior que s. Percebe-se a melhora do modelo a medida em que a curva de elevação aproxima-se da curva ideal (TUFFÉRY, 2011). Na Figura 6 é demonstrado um exemplo da aplicação da curva lift

Figura 6 – Exemplo da curva lift

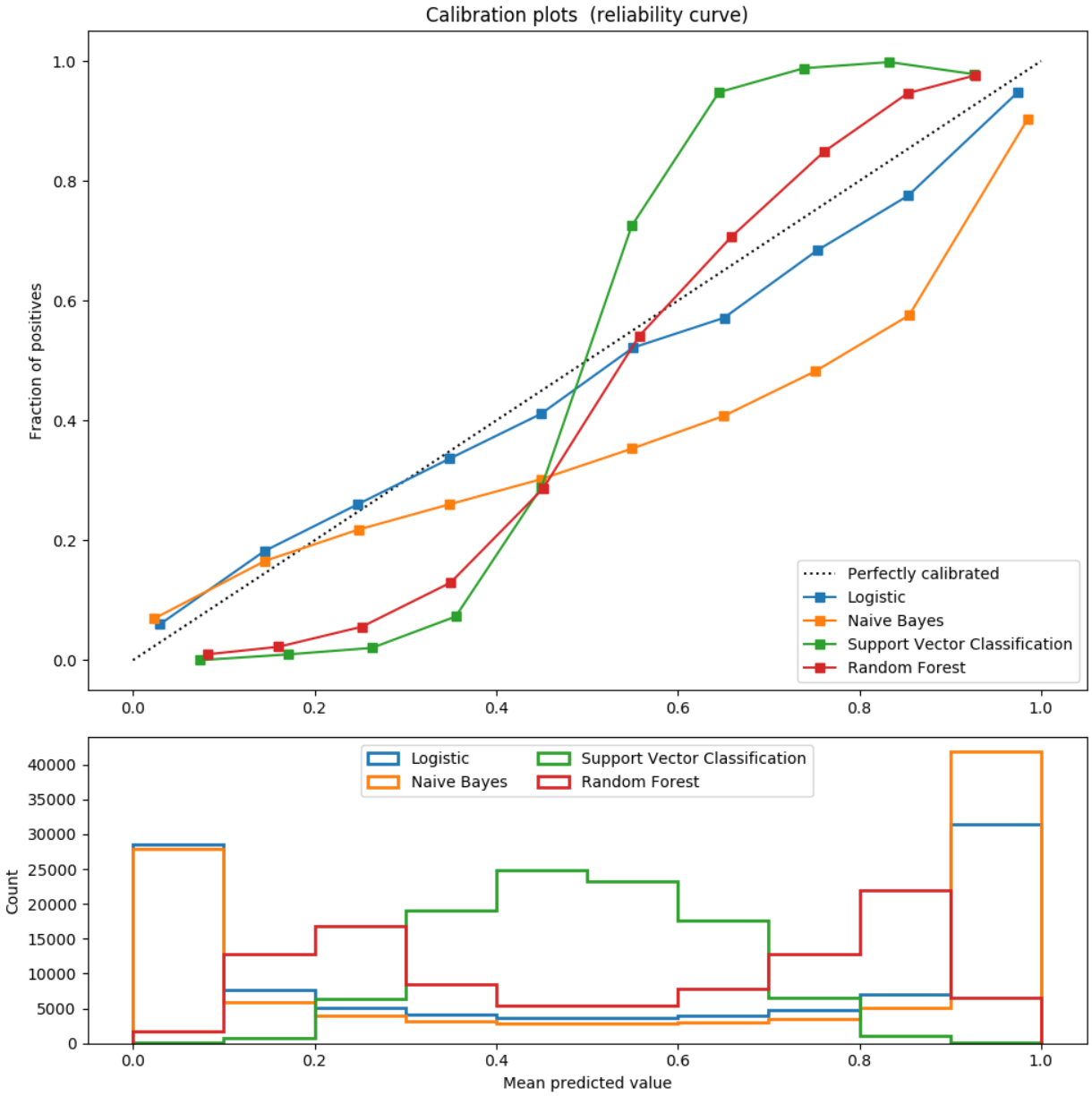


Fonte: Wikimedia (2006)

2.9 CURVA DE CALIBRAÇÃO

A curva de calibração demonstra a correlação da previsão de probabilidades dos classificadores e a probabilidade da classe real. O gráfico de calibração demonstra a probabilidade de uma determinada classe em relação as classes previstas pelos classificadores. A curva de calibração é diferente da curva ROC e Lift anteriormente apresentadas, pois, não se trata de um calibrador probabilístico (VUK; CURK, 2006). Podemos ver um exemplo da aplicação da curva de calibração na Figura 7 a aplicação da curva de calibração demonstrada utiliza-se dos mesmos algoritmos utilizados neste trabalho, porém não se trata da curva de calibração do modelo que será desenvolvido.

Figura 7 – Exemplo da curva de calibração



Fonte: SKLearn (2015)

2.10 MATRIZ DE CONFUSÃO

Para que seja avaliado o comportamento de um dado classificador é preciso analisar a confusão entre as classes. Para tal finalidade uma das ferramentas mais utilizadas é a matriz de confusão ou matriz de erro. Para um problema envolvendo n classes é criada uma matriz $n \times n$ onde as colunas são rotuladas com a classe real e as linhas são rotuladas com as classes previstas. Para cada instância do conjunto de testes há um rótulo na classe real e um rótulo previsto pelo modelo. Assim, a partir da combinação destes dois rótulos é determinado a posição na matriz para a contagem, deixando de forma visível qual classe esta

sendo confundida com outra (PROVOST; FAWCETT, 2013). Podemos observar um exemplo de uma matriz de confusão na Tabela 1. Neste exemplo tem-se duas classes: *Cat* e *Dog*. Na diagonal principal da matriz está contabilizada a quantidade de instâncias de teste (*Classes Reais*) que foram classificadas corretamente pelo classificador (*Classes Preditas*). Nas colunas demonstra-se a quantidade de falsos positivos classificados erroneamente pelo classificador (*Preditas*). Nas linhas demonstra-se a quantidade de falsos negativos classificados erroneamente pelo classificador.

Tabela 1 – Matriz de Confusão

		Classes Preditas	
		Cat	Dog
Classes Reais	Cat	15	35
	Dog	40	10

Fonte: Autor (2021)

3 METODOLOGIA

Neste capítulo são descritos os métodos utilizados para obtenção, limpeza e formatação dos dados. Tais dados servirão de entrada para o modelo de aprendizado de máquina proposto. Também são apresentados nesse capítulo os aspectos de implementação computacional do modelo de recomendação proposto.

3.1 DELINEAMENTO DA PESQUISA

A pesquisa baseia-se nos registros históricos de uma empresa de transporte coletivo público que agrega dados estatísticos de ocupação de ônibus transportando passageiros no município de Toledo-Paraná. Analisando os dados coletados podemos perceber uma ociosidade muito grande do sistema em vários horários do dia, acarretando diversos problemas devido a uma oferta maior de vagas no sistema de transporte coletivo urbano. Ainda, a análise dos dados evidenciou que, para algumas localidades, existe uma oferta menor de vagas do que a necessária para suprir as necessidades reais daquela região, ocasionando assim, em alguns horários, uma superlotação dos veículos.

A pesquisa tem o objetivo de gerar um modelo de recomendação de veículos para atender as demandas da melhor maneira possível, e evitar problemas muito comum em várias cidades que são a lotação de veículos do transporte coletivo.

3.2 COLETA E TRATAMENTO DE DADOS

3.2.1 Extração de quantidade de passageiros

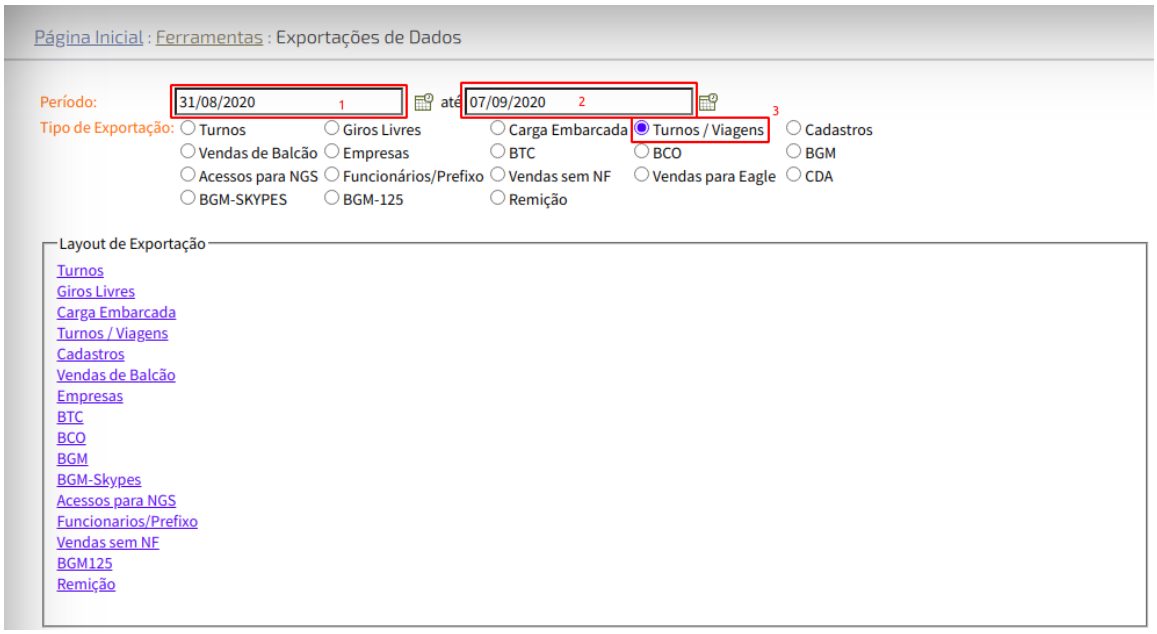
Os dados foram coletados através do sistema de bilhetagem eletrônica utilizado na empresa. Inicialmente, o processo de coleta baseou-se em documentos XML, os quais são fornecidos diretamente pelo sistema Web TDMax Gerencial da empresa Transdata Smart. O período de coleta de dados compreendeu de 15 de dezembro de 2017 a 31 de agosto de 2019.

Esse método de extração baseado em documentos XML foi adotado em primeiro momento pois era a única forma disponibilizada pela empresa desenvolvedora do sistema. Contudo, após algumas negociações, foi disponibilizado acesso direto ao banco de dados. Isso possibilitou aumentar consideravelmente a volume de dados extraídos e um volume expressivo de dados para treinamento era essencial para o presente trabalho. A Figura 8 apresenta a tela do sistema que possibilita a extração dos dados de interesse.

Os dados coletados inicialmente eram de 1 de janeiro de 2014 até 31 de agosto de 2019. Porém, os dados até o dia 14 de dezembro de 2017 apresentavam uma distorção significativa na quantidade de passageiros transportados, pois até tal data o terminal urbano não estava operando, obrigando os usuários a realizarem o embarque no ônibus pela porta

traseira, não contabilizando assim o embarque dessas pessoas nos veículos que as levavam até o destino. Noutras palavras, esse passageiro somente era contabilizado no embarque do bairro até o terminal e nunca do terminal para o bairro. Posteriormente, a implementação da integração proporcionou a abertura do terminal. A partir dessa data o embarque de todas as pessoas começou ocorrer sempre pela porta da frente, possibilitando assim uma contagem dos passageiros totais no veículo para aquela viagem.

Figura 8 – Tela para extração dos dados.



Fonte: Autor (2020)

3.2.2 Extração da capacidade de passageiros dos veículos

Para mensurar a capacidade dos veículos foram enumerados os bancos disponíveis e considerada a área disponível do veiculo em m^2. Para estimar a capacidade total de passageiros foram utilizados critérios adotados pelas cidades brasileiras São Paulo-São Paulo e Curitiba-Paraná, reconhecidas pela forte regulamentação no transporte coletivo urbano. Os aspectos técnicos do transporte em Curitiba são regidos pela lei municipal 12597/08, sendo gerenciados pela e auditados pela URBS (Urbanização de Curitiba S/A). Já na cidade de São Paulo esse papel é da autarquia municipal SPTrans (São Paulo Transporte). Ambas cidades determinam uma proporção de 6 passageiros em pé por m^2. Neste trabalho, considerando uma margem de segurança na indicação do veículo, assumiremos a razão de 5 passageiros por m^2.

As medições foram realizadas por modelo e ano do veiculo, visto que a área de todos os carros daquele ano e modelo tem a mesma quantidade de m^2 e de assentos. Foram extraídos os dados da capacidade total de passageiros dos 54 veículos que compõem a frota da empresa para este estudo.

3.2.3 Perfil do conjunto de dados XML

O conjunto de dados gerado em XML através do Sistema de Bilhetagem Eletrônica (SBE) contém 3 separações, abordadas a seguir. Na Figura 9 é demonstrado parcialmente um documento XML gerado pelo sistema SBE.

Figura 9 – Dataset XML - Dados utilizados no projeto.

```xml
<?xml version="1.0" standalone="yes"?>
<DSExpTurnosViagens xmlns="http://tempuri.org/DSExpTurnosViagens.xsd">
  <Turnos>
    <TurnoID></TurnoID>
    <DataIni></DataIni>
    <DataFim></DataFim>
    <CobradorID></CobradorID>
    <Linha></Linha>
    <Prefixo></Prefixo>
    <NroViagens></NroViagens>
    <Viagens>
      <TurnoID></TurnoID>
      <ViagemID></ViagemID>
      <Sentido></Sentido>
      <DataInicio></DataInicio>
      <DataFim></DataFim>
      <PassageirosQtd><!-- Soma Viagens_qtd --></PassageirosQtd>
      <Viagens_qtd>
        <TurnoID></TurnoID>
        <ViagemID></ViagemID>
        <ProdutoID></ProdutoID>
        <Produto></Produto>
        <Tarifa></Tarifa>
        <FamiliaID></FamiliaID>
        <Familia></Familia>
        <Quantidade></Quantidade>
        <ValorUnitario></ValorUnitario>
        <Contador></Contador>
        <FatorEquivalencia></FatorEquivalencia>
      </Viagens_qtd>
    </Viagens>
  </Turnos>
</DSExpTurnosViagens>
```

Fonte: Autor (2020)

3.2.3.1 *Turnos*

O campo Turnos neste conjunto de dados é a informação principal onde se localizam as seguintes informações:

1. TurnoID: Campo que comtem o ID do turno;
2. TipoTurnoID: Campo que comtem o ID do tipo de turno sendo eles 0 para urbano e 2 para seccionado;
3. DataIni: Campo que comtem a data e hora de inicio do turno aberto pelo motorista;
4. DataFim: Campo que comtem a data e hora de fechamento do turno pelo motorista;
5. EmpresaID: ID da empresa no SBE;
6. NomeFantasia: Nome da empresa no SBE;
7. CobradorID: ID do cobrador responsável por aquele turno;
8. MatriculaCob: Numero da matricula do cobrador na empresa responsável por aquele turno;
9. MotoristaID: ID do motorista responsável por aquele turno;
10. MatriculaMot: Numero da matricula do motorista na empresa responsável por aquele turno;

11. LinhaID: ID da linha no SBE;
12. Linha: Código da linha no sistema urbano (código conhecido pela população que utiliza as linhas);
13. Prefixo: Prefixo do veiculo que está fazendo aquele turno;
14. PrefixoLivre: Novamente se trata do prefixo do veiculo que está fazendo aquele turno;
15. CatracaIni: Contagem inicial da catraca na abertura do turno;
16. CatracaFim: Contagem final da catraca no fechamento do turno;
17. Tabela: Tabela de operação do carro no sistema (não aplicavel no sistema de SBE);
18. Km: Km total do turno (não aplicavel no sistema de SBE);
19. NroViagens: Número de viagens realizadas naquele turno;
20. NumSerie: Número de série do validador;
21. Sequencia: Sequencia do turno processado pelo servidor;
22. DataMov: Data do processamento do turno pelo servidor;

3.2.3.2 *Viagens*

O campo `viagens` no conjunto de dados é onde se localiza os dados referentes as viagens realizadas dentro de um turno, dento do SBE uma viagem é o trecho percorrido do ponto inicial (terminal urbano) até o ponto final (bairro), nesta separação do *dataset* localizam-se as seguintes informações:

1. TurnoID: ID do turno ao qual pertence a viagem;
2. ViagemID: ID da viagem, se trata da sequência da viagem Ex. (1, 2, 3...);
3. Sentido: Apresenta qual o sentido da viagem I para ida e V para volta;
4. DataInicio: Data e hora do inicio da viagem;
5. PontoInicio: ID do ponto que iniciou aquela viagem
6. DataFim: Data e hora do fim da viagem;
7. PontoFim: ID do ponto que finalizou aquela viagem;
8. Km: Km total da viagem (não aplicavel no sistema de SBE);
9. PassageirosQtd: Quantidade total de passageiros naquela viagem.

3.2.3.3 *Viagens quantidade*

o campo `viagens quantidade` é identificado no conjunto de dados como Viagens_qtd, neste parte do *dataset* ficam os dados referente a quantidade de pessoas que embarcaram no ônibus:

1. TurnoID: ID do turno a qual pertence esse tipo de cartão embarcado;
2. ViagemID: ID da viagem a qual pertence esse tipo de cartão embarcado;
3. ProdutoID: ID do produto utilizado no embarque
4. Produto: Descrição do produto utilizado no embarque
5. Tarifa: ID da tarifa cobrada no embarque
6. FamiliaID: ID da familia a qual pertence o cartão utilizado no embarque;

7. Familia: Descrição da familia a qual pertence o cartão utilizado no embarque;
8. Quantidade: Quantidade de pessoas embarcadas na viagem com a mesmas especificaçoes de familia, tarifa e produto utilizados;
9. ValorUnitario: Valor da tarifa cobrada no embarque;
10. Contador: Novamente mensura quantas pessoas embarcadas na viagem com a mesmas especificaçoes de familia, tarifa e produto utilizados;
11. FatorEquivalencia: fator utilizado para mensurar a quantidade de passageiros equivalentes pagantes e meia tarifa sendo 1.0 utilizado em tarifa cheia 0.5 utilizado em meia tarifa e 0 para gratuitos

3.2.4 Tratamento dos dados

A fase de tratamento dos dados foi a fase que demandou mais tempo, pois tínhamos que selecionar os dados de forma correta, removendo os dados que são discrepantes ou que não servem para o nosso modelo, podendo levar a um erro nas classificações. Os dados foram tratados da seguinte forma, primeiramente removido informações que não são relevantes para a geração do modelo, foram removidas as seguintes números de linhas:

- Linha 1 - terminal urbano: esta linha era somente utilizada por cobradores, que ficavam na catraca do terminal, os passageiros que passavam por ela não contabilizavam em nenhum carro;
- Linha 40 - granjas e metropolitano Estas linhas dentro da empresa eram utilizadas somente para abrir o turno em validadores. Esses dois grupos não contavam com o sistema de bilhetagem eletrônica, os validadores ficavam nos carros apenas para reutilizar os mesmos em linhas urbanas;
- Linha 999 - Linha utilizada para manutenção dos veiculos e teste em equipamentos, não é acessadapor passageiros;
- Linha 10 - Linha expresso central descontinuada por uma decisão em conjunto da empresa com a prefeitura devido a baixa demanda em Novembro de 2014, porém algumas vezes esta linha foi aberta de forma incorreta pelos motoristas, o que levou ao descarte destes dados;
- Linha 48 - Linha Metropolitana Seccionada, sistema implantado nos veiculos metropolitano para a bilhetagem eletronica, iniciou as operaçoes em 14 de agosto de 2019, dados removidos por não encaixar no escopo do projeto.

Para a melhor qualidade nos dados foram removidas viagens que tenham mais de 40 minutos, visto que, a viagem de maior duração na cidade de Toledo, informado pelo setor operacional da empresa, é de 30 minutos. A margem de 10 minutos foi adicionada devido a possibilidade de algum atraso ocorrer. As viagens com mais de 40 minutos foram descartadas pois poderiam conter a quantidade de passageiros incorretas para aquele trecho.

3.2.5 Extração de dados diretamente do SGBD

Os dados primeiramente foram obtidos diretamente de arquivos XML gerados individualmente a partir do sistema de gestão da empresa. Em tratativas posteriores, a empresa responsável pelo sistema de bilhetagem eletrônica SBE concedeu o acesso direto ao sistema gerenciador de banco de dados (SGBD), possibilitando assim extrair os dados via consultas SQL diretamente da base de dados.

3.2.5.1 *Consulta SQL para extração de dados do SGBD*

A consulta SQL apresentada na Listagem 1 foi criada em substituição ao método citado no item 3.2.1. Conforme já relatado, foi concedido acesso direto a base de dados da empresa SBE, permitindo automatizar e customizar a tarefa de extração de dados. Na consulta foram preservadas as mesmas regras de tratamento de dados definidas para a coleta feita via XML.

Listagem 1 Consulta SQL para extração direta dos dados.

```sql
SELECT
       CAST(T.TurnoID AS varchar) AS TurnoID,
       CAST(V.ViagemID AS varchar) AS ViagemID,
       CAST(L.Codigo AS VARCHAR) AS Codigo,
       CAST(V.Sentido AS VARCHAR) AS Sentido,
       CAST(T.Prefixo AS varchar) AS Prefixo,
       CAST(V.DataInicio AS datetime) AS DataInicio,
       CAST(V.DataFim AS datetime) AS DataFim,
       DATEPART(dw, V.DataInicio) AS DiaDaSemana,
       DATEDIFF(mi, V.DataInicio, V.DataFim) AS Diferenca,
       SUM(VQ.Contador) AS QTDPasageirosNaViagem
FROM dbo.Viagens_qtd VQ
RIGHT JOIN dbo.Viagens V ON V.TurnoID = VQ.TurnoID
                         AND VQ.ViagemID = V.ViagemID
RIGHT JOIN dbo.Turnos T ON T.TurnoID = V.TurnoID
RIGHT JOIN dbo.Linhas L ON T.LinhaID = L.LinhaID
WHERE
      L.Codigo NOT IN('043', '001', '999', '048') AND
      CAST(V.DataInicio AS DATE) BETWEEN '2017-12-15' AND '2020-02-29' AND
      DATEDIFF(mi, V.DataInicio, V.DataFim) BETWEEN 5 AND 50
GROUP BY
         T.TurnoID,
         V.ViagemID,
         L.Codigo,
         V.Sentido,
         V.DataInicio,
         V.DataFim,
         T.Prefixo
ORDER BY
         T.TurnoID,
         V.ViagemID;
```

Fonte: Autor (2021)

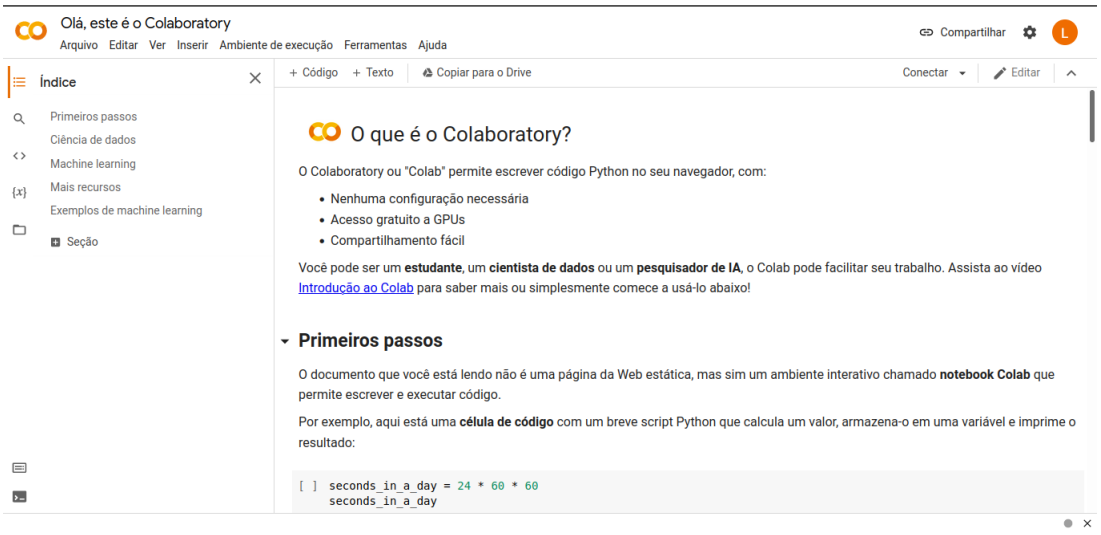
3.3 CRIAÇÃO DO ALGORITMO

Nesta seção será abordado todo o processo e ferramentas utilizadas para a elaboração do modelo.

3.3.1 Google colab

O Google Colaboratory (Colab) é um ambiente de notebooks Jupyter que não requer configuração externa prévia e é executado diretamente em máquinas virtuais Google na nuvem. Através do Colab é possível gratuitamente executar códigos em Python mesclados com texto/gráficos e acessar recursos de computação científica, *hardware* especializado como GPUs e TPUs, diretamente no navegador Web. Para adicionar conteúdo existem dois tipos de campo de entrada (célula ou *cell*): código ou texto. Na Figura 10 é demonstrada a tela inicial do Colab após iniciar o projeto. Na área de aprendizado de máquina e visão computacional o Colab já conta com centenas de bibliotecas pré-instaladas, dentre as quais as populares Numpy, Pandas, Matplotlib, Scikit-learn, Keras e TensorFlow. Além das bibliotecas pré-instaladas também suporta a instalação de pacotes de bibliotecas externas. O Colab atualmente permanece com acesso gratuito, porém existe a possibilidade de contratar planos pagos que oferecem melhores GPUs, mais memória RAM e mais espaço de armazenamento.

Figura 10 – Interface do Google Colab.



Fonte: Autor (2021)

3.3.2 Bibliotecas utilizadas no projeto

A Listagem 2 apresenta uma listagem das bibliotecas Python importadas e usadas no decorrer do trabalho. Destacam-se as bibliotecas (a) Pandas, para manipular e visualizar conjunto de dados, em especial tabelas numéricas e séries temporais; (b) Scikit-learn para criação

dos modelos de aprendizado de máquina; (c) Numpy para lidar com operações matemáticas em matrizes multidimensionais e (c) Matplotlib para plotar os gráficos.

Listagem 2 Bibliotecas Python Utilizadas

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import math
5  from datetime import datetime
6  from sklearn.model_selection import train_test_split
7  from sklearn import svm
8  from sklearn.pipeline import make_pipeline
9  from sklearn.preprocessing import StandardScaler
10 from sklearn.metrics import confusion_matrix,accuracy_score
11 from sklearn.metrics import roc_curve, auc, recall_score,precision_score
12 from sklearn.metrics import  multilabel_confusion_matrix, plot_confusion_matrix
13 from sklearn.naive_bayes import GaussianNB
14 from sklearn.calibration import calibration_curve, CalibratedClassifierCV
15 from sklearn.linear_model import LogisticRegression
16 from sklearn.ensemble import RandomForestClassifier
17 from sklearn.datasets import make_classification
18 from sklearn.model_selection import cross_val_score
```

Fonte: Autor (2021)

3.3.3 Importação e manipulação do CSV

Como mencionado anteriormente, utilizamos a biblioteca pandas, para importar e manipular a aquivo CSV gerado a partir da consulta SQL. Na Listagem 3 é mostrado o código utilizado para realizar a leitura do CSV com o pandas, neste caso, passamos dois parâmetros o local do arquivo e o separador usado no CSV, vale ressaltar que essa biblioteca permite mais parâmetros e mais tipos de arquivos.

Listagem 3 Importação de Arquivo CSV

```
1  data = pd.read_csv('/content/drive/MyDrive/TCC/Dados/TCC_DADOS.csv', sep=";")
```

Fonte: Autor (2021)

Após a importação do arquivo CSV iniciamos a manipulação do mesmo como demonstrado na Listagem 4. Nesta manipulação inicial foram inserido 6 novos campos no *dataset*, utilizando dados já existentes apenas modificando a forma de visualização do mesmo, bem como, foram modificados 2 parâmetros.

Os valores alterados foram os seguintes:

1. QTDPassageirosNaViagem: Neste campo fois feito a seguinte modificação, todos os campos do CSV qeu vieram como NULL no dataset viraram NaN, com o código da linha 1 na Figura **??**, encontramos os valores qeu estão como NaN e alterams para 0.

Listagem 4 Manipulação de Dataset

```
data['QTDPasageirosNaViagem'] = data['QTDPasageirosNaViagem'].replace(np.nan, 0)
data['Sentido'] = data['Sentido'].replace('V', 0)
data['Sentido'] = data['Sentido'].replace('I', 1)
data['Ano'] = pd.DatetimeIndex(data['DataInicio']).year
data['Mes'] = pd.DatetimeIndex(data['DataInicio']).month
data['Dia'] = pd.DatetimeIndex(data['DataInicio']).day
data['Hora'] = pd.DatetimeIndex(data['DataInicio']).hour
data['Minuto'] = pd.DatetimeIndex(data['DataInicio']).minute
data['Segundo'] = pd.DatetimeIndex(data['DataInicio']).second
```

Fonte: Autor (2021)

2. Sentido: O campo sentido inicialmente estava preenchido com as letras I representando o setido de ida e com a letra V representando o sentido de volta, estes valores foram alterados para 1 e 0 respectivamente, essa alteração foi necessaria para a aplicação dos algoritmos.

Já os seis campos adicionados tratam-se da transformação do campo `DataInicio`, expandido para os campos `Dia`, `Mes`, `Ano`, `Hora`, `Minuto` e Segundos no dataset.

Na Figura 5 é apresentado a função criada para fazer a tipagem correta do rótulo do veiculo, levando em consideração a quantidade de passageiros que estiveram no veiculo no decorrer da viagem, na defesa de TCC 1 houve uma indagação da banca, sobre tentar identificar o desembarque das pessoas no decorrer da viagem para obter esse numero de uma forma mais precisa, no decorrer deste trabalho, levantamos hipóteses de formas para conseguir esses números, para a resolução deste problema precisaria do desenvolvimento de um hardware para essa contagem, oque fugiria do escopo da proposta.

Optamos então pela seguinte métrica, considerar somente 75%, dos passageiros embarcados no veiculo no decorrer da viagem, dessa form chegamos a função apresentada abaixo na Figura 5, onde inteiramos o dataset para obter o dado de quantidade de passageiros na viagem, e posteriormente definir qual o rótulo correto, de 0 até 30 passageiros é Micro Onibus, de 30 até 70 passageiros é Midi Onibus, e mais de 70 recomenda-se a categoria Padron.

Após a rotulagem do dataset, foi criado uma função para atribuir uma faixa de horário no dataset, informação que tem uma grande importância no fluxo de pessoas no transporte coletivo, a Listagem 6 mostra a implementação desta função com algumas variáveis a menos para não ficar extenso, o código completo encontra-se no Apêndice XX.

Listagem 5 Rotina para tratamento dos dados de lotação do veiculo.

```
def tipoDeVeiculo(dataframe):
  passageiros = []
  tipoDeVeiculoRotulo = []
  for index, row  in dataframe.iterrows():
    if row['QTDPasageirosNaViagem'] != 0:
      QTDPassageiros75PorCento = np.ceil(row['QTDPasageirosNaViagem'] * 0.75)
      passageiros.append(QTDPassageiros75PorCento)
      if QTDPassageiros75PorCento > 0 and QTDPassageiros75PorCento <  30:
        tipoDeVeiculoRotulo.append("Micro Onibus")
      elif QTDPassageiros75PorCento >= 30 and QTDPassageiros75PorCento <=  70:
        tipoDeVeiculoRotulo.append("Midi Onibus")
      else:
        tipoDeVeiculoRotulo.append("Padron")
    else:
      passageiros.append(0)
      tipoDeVeiculoRotulo.append("Micro Onibus")

  return tipoDeVeiculoRotulo, passageiros
```

Fonte: Autor (2021)

Listagem 6 – Rotina para obter as faixas de horários.

```
def manipularDatas(dataframe):
  faixaDeHorario = []
  meiaNoite = datetime.now().replace(hour=0,minute=0,second=0,microsecond=0).time()
  cincoDaManha = datetime.now().replace(hour=5,minute=0,second=0,microsecond=0).time()
  seisDaManha = datetime.now().replace(hour=6,minute=0,second=0,microsecond=0).time()
  seteDaManha = datetime.now().replace(hour=7,minute=0,second=0,microsecond=0).time()
  oitoDaManha = datetime.now().replace(hour=8,minute=0,second=0,microsecond=0).time()
  noveDaManha = datetime.now().replace(hour=9,minute=0,second=0,microsecond=0).time()
  dezDaManha = datetime.now().replace(hour=10,minute=0,second=0,microsecond=0).time()
  onzeDaManha = datetime.now().replace(hour=11,minute=0,second=0,microsecond=0).time()
  meioDia = datetime.now().replace(hour=12,minute=0,second=0,microsecond=0).time()
  umaDaTarde = datetime.now().replace(hour=13,minute=0,second=0,microsecond=0).time()
  duasDaTarde = datetime.now().replace(hour=14,minute=0,second=0,microsecond=0).time()
  tresDaTarde = datetime.now().replace(hour=15,minute=0,second=0,microsecond=0).time()
  quatroDaTarde = datetime.now().replace(hour=16,minute=0,second=0,microsecond=0).time()
  cincoDaTarde = datetime.now().replace(hour=17,minute=0,second=0,microsecond=0).time()
  seisDaTarde = datetime.now().replace(hour=18,minute=0,second=0,microsecond=0).time()
  seteDaNoite = datetime.now().replace(hour=19,minute=0,second=0,microsecond=0).time()
  oitoDaNoite = datetime.now().replace(hour=20,minute=0,second=0,microsecond=0).time()
  noveDaNoite = datetime.now().replace(hour=21,minute=0,second=0,microsecond=0).time()
  dezDaNoite = datetime.now().replace(hour=22,minute=0,second=0,microsecond=0).time()
  onzeDaNoite = datetime.now().replace(hour=23,minute=0,second=0,microsecond=0).time()
  for index, row  in dataframe.iterrows():
```

```
    horario = pd.to_datetime(row['DataInicio']).time()
    if horario >= cincoDaManha and horario < seisDaManha:
      faixaDeHorario.append('Das 05:00 às 06:00')
    elif  horario >= seisDaManha and horario < seteDaManha:
      faixaDeHorario.append('Das 06:00 às 07:00')
    elif  horario >= seteDaManha and horario < oitoDaManha:
      faixaDeHorario.append('Das 07:00 às 08:00')
    elif  horario >= oitoDaManha and horario < noveDaManha:
      faixaDeHorario.append('Das 08:00 às 09:00')
    elif  horario >= noveDaManha and horario < dezDaManha:
      faixaDeHorario.append('Das 09:00 às 10:00')
    elif  horario >= dezDaManha and horario < onzeDaManha:
      faixaDeHorario.append('Das 10:00 às 11:00')
    elif  horario >= onzeDaManha and horario < meioDia:
      faixaDeHorario.append('Das 11:00 às 12:00')
    elif  horario >= meioDia and horario < umaDaTarde:
      faixaDeHorario.append('Das 12:00 às 13:00')
    elif  horario >= umaDaTarde and horario < duasDaTarde:
      faixaDeHorario.append('Das 13:00 às 14:00')
    elif  horario >= duasDaTarde and horario < tresDaTarde:
      faixaDeHorario.append('Das 14:00 às 15:00')
    elif  horario >= tresDaTarde and horario < quatroDaTarde:
      faixaDeHorario.append('Das 15:00 às 16:00')
    elif  horario >= quatroDaTarde and horario < cincoDaTarde:
      faixaDeHorario.append('Das 16:00 às 17:00')
    elif  horario >= cincoDaTarde and horario < seisDaTarde:
      faixaDeHorario.append('Das 17:00 às 18:00')
    elif  horario >= seisDaTarde and horario < seteDaNoite:
      faixaDeHorario.append('Das 18:00 às 19:00')
    elif  horario >= seteDaNoite and horario < oitoDaNoite:
      faixaDeHorario.append('Das 19:00 às 20:00')
    elif  horario >= oitoDaNoite and horario < noveDaNoite:
      faixaDeHorario.append('Das 20:00 às 21:00')
    elif  horario >= noveDaNoite and horario < dezDaNoite:
      faixaDeHorario.append('Das 21:00 às 22:00')
    elif  horario >= dezDaNoite and horario < onzeDaNoite:
      faixaDeHorario.append('Das 22:00 às 23:00')
    elif  horario >= onzeDaNoite and horario < meiaNoite:
      faixaDeHorario.append('Das 23:00 às 00:00')
    else:
      faixaDeHorario.append('Outras - Especial')

  return faixaDeHorario
```

Fonte: Autor (2021)

Na sequencia foi realizada as seguintes alterações como demonstramos na Figura 7,

atribuído o campo faixa de horário que é retornado pela função demonstrada na Figura 6, o mesmo ocorre com o campo Rotulo e QTDPassageirosLimiar, os valores atribuídos a esse campo são os valores retornados pela função demonstrada na Figura 5, após a atribuição destes campos mencionados acima, novamente faremos a Substituição dos campos textuais por valores numéricos, para que seja possível a análise através dos algoritmos.

Listagem 7 Manipulação do dataset e inserção de faixas de horários e rótulos

```
data['FaixaDeHorario'] = faixaDeHorario
data['QTDPassageirosLimiar'] = passageiros
data['Rotulo'] = rotulo
data['Rotulo'] = data['Rotulo'].replace("Micro Onibus", 0)
data['Rotulo'] = data['Rotulo'].replace("Midi Onibus", 1)
data['Rotulo'] = data['Rotulo'].replace("Padron", 2)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 05:00 às 06:00",0)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 06:00 às 07:00",1)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 07:00 às 08:00",2)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 08:00 às 09:00",3)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 09:00 às 10:00",4)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 10:00 às 11:00",5)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 11:00 às 12:00",6)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 12:00 às 13:00",7)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 13:00 às 14:00",8)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 14:00 às 15:00",9)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 15:00 às 16:00",10)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 16:00 às 17:00",11)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 17:00 às 18:00",12)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 18:00 às 19:00",13)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 19:00 às 20:00",14)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 20:00 às 21:00",15)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 21:00 às 22:00",16)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 22:00 às 23:00",17)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Das 23:00 às 00:00",18)
data['FaixaDeHorario'] = data['FaixaDeHorario'].replace("Outras - Especial",19)
```

Fonte: Autor (2021)

3.3.4 Extração de descritores

Na Listagem 8 tem-se a extração dos descritores, passo anterior a separação do conjunto de dados em instâncias de treinamento e teste. Nesta etapa retiramos os dados que não serão repassados ao modelo, isto é, os campos `QTDPasageirosNaViagem`, `Rotulo`, `DataInicio` e `DataFim`. Tais campos foram removidos por não agregarem relevância ao modelo. O campo `QTDPasageirosNaViagem` foi removido pois o campo `QTDPassageirosLimiar` ocupou seu lugar, conforme já mostrado na Listagem 6.

Listagem 8 Extração de descritores

```
descritores = data.drop(columns=[
'QTDPasageirosNaViagem',
'Rotulo', 'DataInicio',
'DataFim'
])
```

Fonte: Autor (2021)

3.3.5 Conjunto de treinamento e teste

A Figura 9 apresenta o código para dividir os dados em dois subconjuntos disjuntos: treinamento e teste. Utilizamos a função `train_test_split()` do módulo `model_selection` da biblioteca sklearn, definindo 50% do *dataset* para treino e 50% para testes. A saída desta função são os subconjuntos, onde `X_train` representa os descritores de treinamento, `X_test` representa os descritores utilizados para teste e validar os modelos, `y_train` são os rótulos utilizados para treinar o modelo, `y_train` são os rótulos utilizados para a validação do modelo.

Listagem 9 Split de dados de treino e teste.

```
X_train, X_test, y_train, y_test = train_test_split(descritores, data['Rotulo'],
test_size=0.5, random_state=0)
```

Fonte: Autor (2021)

3.3.6 *Naïve Bayes*

Conforme observa-se na Listagem 10 um dos algoritmos utilizados neste trabalho é o *Naïve Bayes*, seguindo a documentação da biblioteca Sklearn. Na primeira linha da Listagem 10 é instanciada uma nova classe do GausianNB. Em seguida iniciamos o treinamento do modelo através do método `fit()` passando os parâmetros `X_train` e `y_train`, isto é, os descritores e respectivos rótulos das instâncias do conjunto de treinamento. Após o treinamento do modelo através do algoritmo, incia-se a fase de teste. Utilizamos o método `predict()`, passando o parâmetro o `X_test`. A predição retorna uma lista de classes preditas que são comparadas com o conjunto `y_test` e os acertos são computados para avaliar a assertividade do modelo.

Listagem 10 Algoritmo de Naive Bayes

```
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train)
predicao = gnb.predict(X_test)
```

Fonte: Autor (2021)

3.3.7 Regressão Logistica

A Listagem 11 apresenta a aplicação do algoritmo Regressão Logística. Primeiramente instanciamos a classe `LogisticRegression()` com os parâmetros `random_state=0` e `max_iter=500`. As fases seguintes de treinamento, predição e avaliação dos resultados do modelo segue o descrito na Seção 3.3.6.

Listagem 11 Algoritmo de Regressão Legística

```
1 rfc = RandomForestClassifier(max_depth=4, random_state=2)
2 rfc.fit(X_train, y_train)
3 predicao_RF = rfc.predict(X_test)
```

Fonte: Autor (2021)

3.3.8 *Random Forest*

A Listagem 12 apresenta a aplicação do algoritmo *Random Forest*. Primeiramente instanciamos a classe `RandomForestClassifier()`. São usados os parâmetros `max_depth=4` e `random_state=2`. As fases seguintes de treinamento, predição e avaliação dos resultados do modelo segue o descrito na Seção 3.3.6.

Listagem 12 Algoritmo de Random Forest

```
1 rfc = RandomForestClassifier(max_depth=4, random_state=2)
2 rfc.fit(X_train, y_train)
3 predicao_RF = rfc.predict(X_test)
```

Fonte: Autor (2021)

3.3.9 Svm

A Listagem 13 apresenta a aplicação do algoritmo *SVM*. A implementação do SVM diferente das anteriores, pois baseai-se em duas classes dentro de uma *pipeline*. A função `make_pipeline()` recebe dois parâmetros, o primeiro é a classe `StandardScaler()`, um pré processador que padroniza os dados comumente usado quando há muitos descritores; e a própria classe do algoritmo *SVM*. A classe recebe um parâmetro que é a função de decisão que será utilizada. As fases seguintes de treinamento, predição e avaliação dos resultados do modelo segue o descrito na Seção 3.3.6.

3.4 MÉTRICAS DE AVALIAÇÃO

Nesta seção são apresentadas as funções utilizadas para gerar a matriz de confusão e a curva ROC.

Listagem 13 Algoritmo de SVM

```
clf = make_pipeline(StandardScaler(), svm.SVC(decision_function_shape='ovo'))
clf.fit(X_train, y_train)
predicao_SVM = clf.predict(X_test)
```

Fonte: Autor (2021)

3.4.1 Matriz de confusão

Na Listagem 14 é mostrada a obtenção das métricas `falso positivos`, `falso negativos`, `verdadeiro positivos` e `verdadeiro negativos`. Tais métricas são essenciais para a produção da matriz de confusão.

Listagem 14 Função de matriz de confusão - Google Colab

```
def getConfusionMatrixMetrics(test_values, predict_valoues):
  arrayMC = multilabel_confusion_matrix(test_values, predict_valoues)
  TP = arrayMC[:, 1,1]
  TN = arrayMC[:, 0,0]
  FN = arrayMC[:, 1,0]
  FP = arrayMC[:, 0,1]

  return TP, TN, FN, FP
```

Fonte: Autor (2021)

3.4.2 Testes de diagnóstico

Na Listagem 15 é demonstrado a função criada para extrair: a taxa de verdadeiro positivo (TPR), taxa de verdadeiro positivo (TNR), precisão, valor predito negativo (NPV), taxa de falso positivo (FPR), taxa de falso negativo (FNR), taxa de descoberta falsa (FDR), taxa de falsa omissão (FOR), limite de prevalência (PT), revocação (*recall*), F1-Score, acurácia (*accuracy*) e acurácia balanceada (*balanced accuracy*. Tais métricas são utilizadas para avaliar cada modelo de classificação, no capítulo 4.

Listagem 15 Teste de Diagnostico

```
def getTrueTableMetrics(TP, TN, FN, FP, class_position):
  # Sensitivity, hit rate, recall, or true positive rate
  TPR = TP[class_position]/(TP[class_position]+FN[[class_position]])
  # Specificity or true negative rate
  TNR = TN[class_position]/(TN[class_position]+FP[class_position])
  # Precision or positive predictive value
  PPV = TP[class_position]/(TP[class_position]+FP[class_position])
  # Negative predictive value
  NPV = TN[class_position]/(TN[class_position]+FN[class_position])
  # Fall out or false positive rate
  FPR = FP[class_position]/(FP[class_position]+TN[class_position])
  # False negative rate
  FNR = FN[class_position]/(TP[class_position]+FN[class_position])
  # False discovery rate
  FDR = FP[class_position]/(TP[class_position]+FP[class_position])
  #false omission rate
  FOR = (FN[class_position]/(FN[class_position]+TN[class_position]))
  #prevalence threshold
  PT = ((math.sqrt((TPR*FPR)) - FPR) /(TPR-FPR))
  #Precision
  PRECISION = (TP[class_position]/(TP[class_position]+FP[class_position]))
  #Recall
  RECALL = (TP[class_position]/(TP[class_position]+FN[class_position]))
  #F1-Score
  F1 = ((2*PRECISION*RECALL)/(PRECISION+RECALL))

  # Overall accuracy
  ACC = (TP[class_position]+TN[class_position])/(TP[class_position]+FP[class_position]+FN[class_positi

  # Balanced accuracy
  BA = (TPR+TNR)/2

  return (TPR,TNR, PPV,NPV,FPR,FNR,FDR, ACC, BA[0], FOR, PT, PRECISION, RECALL, F1)
```

Fonte: Autor (2021)

4 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados obtidos com a aplicação dos quatro algoritmos selecionados: *Naïve Bayes*, Regressão Logística, *Random Forest* e *Support Vector Machine* (SVM). Para a execução de cada algoritmo foram registradas cinco métricas de avaliação: revocação (*recall*), precisão (*precision*), F1-score, acurácia (*accuracy*) e acurácia balanceada (*balanced accuracy*). Em cada uma das seções será apresentada a média dos indicadores em cada classe e na sequência o resultado compilado do modelo. Também é apresentada a matriz de confusão para cada modelo. A métrica escolhida para a avaliação do modelo que foi a F1-Score. A métrica do F1-Score foi escolhida para a avaliação, visto que, ela combina o resultado da métrica de revocação (*recall*), que nos diz com qual frequência o classificador encontra exemplos da classe e se realmente for dela ele o classifica, e combina também o resultado da métrica precisão (*precision*), que nos demonstra da classe que eu indiquei quais estavem realmente certos, a métrica do F1-Score é a média harmônica destas métricas, além de ter a aplicabilidade, visto que, temos classes desproporcionais

4.1 NAÏVE BAYES

Na Tabela 2 são apresentados cinco parâmetros que podem ser utilizados para a avaliação do modelo.

Tabela 2 – Resultado Naïve Bayes.

Classe	Recall	Precision	F1-Score	Accuracy	Balanced Accuracy
Micro Onibus	97,16%	99,89%	98,51%	97,26%	97,89%
Midi Onibus	98,24%	71,71%	82,91%	97,23%	97,70%
Padron	100,00%	13,26%	23,42%	99,97%	99,98%

Fonte: Autor (2021)

Neste primeiro algoritmo notou-se os indicadores excelentes para a previsão da classe `micro ônibus`, porém notamos que há uma queda grande na métrica F1-Score na classe de `midi ônibus` e novamente há uma queda do F1-Score na classe `padron`.

Na Tabela 3 apresentamos os dados compilados do modelo completo. Podemos observar um F1-Score utilizado na avaliação do modelo com uma média de 68,28%, as outras métricas nitidamente apresentam pouca mudança no geral, enquanto o F1-Score se trata de uma excelente métrica para essa avaliação de qualidade.

Na matriz de confusão apresentada na Tabela 4, pode-se observar os seguinte: na classe `micro ônibus` obtivemos 296331 verdadeiros positivos, 21954 verdadeiros negativos, 8737 falsos negativos e 308 falsos positivos. Na classe `midi ônibus` obtivemos 21941 verdadeiros positivos, 296344 verdadeiros negativos, 8737 falsos negativos e 308 falsos positivos. Na classe

Tabela 3 – Resultados compilados para todo o modelo.

Classe	Recall	Precision	F1-Score	Accuracy	Balanced Accuracy
Geral	98,47%	61,62%	68,28%	98,15%	98,52%

Fonte: Autor (2021)

padron obtivemos 13 verdadeiros positivos, 318272 verdadeiros negativos, 8737 falsos negativos e 308 falsos positivos.

Tabela 4 – Matriz de Confusão Naïve Bayes

		Classes Preditas		
		Micro	Midi	Padron
Classes Reais	Micro	97,16%	2,84%	0,00%
	Midi	1,38%	98,24%	0,38%
	Padron	0,00%	0,00%	100%

Fonte: Autor (2021)

4.2 REGRESSÃO LOGÍSTICA

Na Tabela 5 estão demonstrados as métricas para o algoritmo Regressão Logística.

Tabela 5 – Resultado Regressão Logística.

Classe	Recall	Precision	F1-Score	Accuracy	Balanced Accuracy
Micro Onibus	99,66%	99,62%	99,64%	99,33%	97,24%
Midi Onibus	94,82%	95,35%	95,08%	99,33%	97,24%
Padron	0,00%	0,00%	0,00%	99,99%	49,99%

Fonte: Autor (2021)

No algoritmo de regressão logística, percebe-se uma melhora em comparação ao algoritmo *Naïve Bayes* para a classe de midi ônibus, porém, vale ressaltar que esse algoritmo teve bastante dificuldade na predição de um veiculo padron. Isso pode der parcialmente

explicado pelo fato da base de dados possuir uma quantidade de classes padron muito menor do que as classes `micro ônibus` e `midi ônibus`.

Na Tabela 6 está demonstrado os dados compilados do modelo inteiro, observa-se o indicador F1-Score com uma média de 64,99%, percebe-se este indicador menor que o indicador demonstrado na sessão anterior do algoritmo *Naïve Bayes*, esta média mais baixa se deu por um problema anteriormente relatado que é do algoritmo não conseguir efetua a previsão correta da categoria padron, deixando assim o indicador F1-Score com uma média menor.

Tabela 6 – Resultados compilados para todo o modelo.

Classe	**Recall**	**Precision**	**F1-Score**	**Accuracy**	**Balanced Accuracy**
Geral	64,83%	64,99%	64,91%	99,55%	81,49%

Fonte: Autor (2021)

Na matriz de confusão apresentada na Tabela 7, podemos observar os seguintes dados obtidos, na classe micro ônibus obtivemos 303962 verdadeiros positivos, 21178 verdadeiros negativos, 1022 falsos negativos, e 1168 falsos positivos. Na classe midi ônibus obtivemos 21178 verdadeiros positivos, 303962 verdadeiros negativos, 1022 falsos negativos, e 1168 falsos positivos. Na classe padron obtivemos 0 verdadeiros positivos, 325140 verdadeiros negativos, 1022 falsos negativos, e 1168 falsos positivos.

Tabela 7 – Matriz de Confusão Regressão Logística

		Classes Preditas		
		Micro	Midi	Padron
Classes Reais	Micro	99,67%	0,33%	0,00%
	Midi	5,17%	94,82%	1,00
	Padron	0,00%	100%	0,00%

Fonte: Autor (2021)

4.3 RANDOM FOREST

Na Tabela 8 estão demonstrados as métricas para o algoritmo *Random Forest*.

Tabela 8 – Resultado Random Forest.

Classe	Recall	Precision	F1-Score	Accuracy	Balanced Accuracy
Micro Onibus	100,00%	99,99%	99,99%	99,99%	99,97%
Midi Onibus	99,95%	99,95%	99,95%	99,99%	99,97%
Padron	0,00%	0,00%	0,00%	99,99%	50,00%

Fonte: Autor (2021)

No algoritmo de *Random forest* podemos notar uma melhora do indicador F1-Score, comparado ao algoritmo de regressão logística, porém novamente é possível notar que o algoritmo não foi capaz de predizer a classe padron, percebe-se que, até o momento somente o algoritmo de *Naïve Bayes* conseguiu, mesmo que com um índice no F1-Score baixo comparado as outra classes preditas por ele.

Na Tabela 9 está demonstrado os dados compilados do modelo inteiro, observa-se o indicador F1-Score com uma média de 66,65%, nota-se uma melhora deste indicador comparado a sessão anterior onde apresentamos o algoritmo de regressão logística, este aumento, se deu devido a melhora da predição do algoritmo na classe `midi ônibus`, esta pequena melhoria serviu para aumentar o indicador F1-Score.

Tabela 9 – Resultados compilados para todo o modelo.

Classe	Recall	Precision	F1-Score	Accuracy	Balanced Accuracy
Geral	66,65%	66,65%	66,65%	99,99%	83,31%

Fonte: Autor (2021)

Na matriz de confusão apresentada na Tabela 10, podemos observar os seguintes dados: para a classe micro ônibus obtivemos 304.983 verdadeiros positivos, 22.325 verdadeiros negativos, nenhum falsos negativo e 22 falsos positivos. Na classe `midi ônibus` obtivemos 22.325 verdadeiros positivos, 304.983 verdadeiros negativos, nenhum falsos negativo e 22 falsos positivos. Na classe padron obtivemos 0 verdadeiros positivos, 327.308 verdadeiros negativos, nenhum falso negativo e 22 falsos positivos.

Tabela 10 – Matriz de Confusão Random Forest

		Classes Preditas		
		Micro	Midi	Padron
Classes Reais	Micro	100%	0,00%	0,00%
	Midi	0,04%	99,96%	0,00%
	Padron	15,38%	84,62%	0,00%

Fonte: Autor (2021)

4.4 SUPPORT VECTOR MACHINE (SVM)

A Tabela 11 exibe os resultados das métricas para o algoritmo *SVM*.

Tabela 11 – Resultado Support Vector Machine.

Classe	**Recall**	**Precision**	**F1-Score**	**Accuracy**	**Balanced Accuracy**
Micro Onibus	99,92%	99,98%	99,95%	99,92%	99,86%
Midi Onibus	99,80%	98,99%	99,39%	99,91%	99,86%
Padron	7,69%	100%	14,28%	99,99%	53,84%

Fonte: Autor (2021)

Nos resultados do algoritmo SVM podemos notar uma leve queda no indicador F1-Score, tanto na classe micro ônibus como `midi ônibus`. Entretanto, podemos perceber que o F1-Score desta vez previu de maneira satisfatória o padron, mesmo o *dataset* possuindo uma pequena quantidade de instâncias dessa classe, sendo este o segundo algoritmo que prevê acertadamente esta categoria de veículo.

A Tabela 12 apresenta os dados compilados do modelo inteiro. Observa-se o indicador F1-Score com uma média de 71,21%, sendo essa a maior média deste indicador dentre os outros três algoritmos apresentados anteriormente. Isso dá-se pelo fato deste algoritmo ter uma excelente porcentagem para indicações da classe `micro ônibus` e `midi ônibus`, além de ser o algoritmo que após o *Naïve Bayes* conseguiu realizar a predição de veículos padron, mesmo a porcentagem dessa previsão não sendo superior ao *Naïve Bayes*. Assim, a média alta deste caso está atrelada a melhoria nas duas classes `midi` e a `micro`.

Tabela 12 – Resultados compilados para todo o modelo.

Classe	Recall	Precision	F1-Score	Accuracy	Balanced Accuracy
Geral	69,14%	99,66%	71,21%	99,94%	84,52%

Fonte: Autor (2021)

Na matriz de confusão apresentada na Tabela 13, podemos observar o seguinte: na classe `micro ônibus` obtivemos 304.762 verdadeiros positivos, 22.291 verdadeiros negativos, 215 falsos negativos e 56 falsos positivos. Na classe `midi ônibus` obtivemos 22.290 verdadeiros positivos, 304.769 verdadeiros negativos, 215 falsos negativos e 56 falsos positivos. Na classe `padron` obtivemos 1 verdadeiro positivo, 327.058 verdadeiros negativos, 215 falsos negativos e 56 falsos positivos.

Tabela 13 – Matriz de Confusão SVM

		Classes Preditas		
		Micro	Midi	Padron
Classes Reais	Micro	99,93%	0,07%	0,00%
	Midi	0,20%	99,80%	0,00%
	Padron	0,00%	92,31%	7,69%

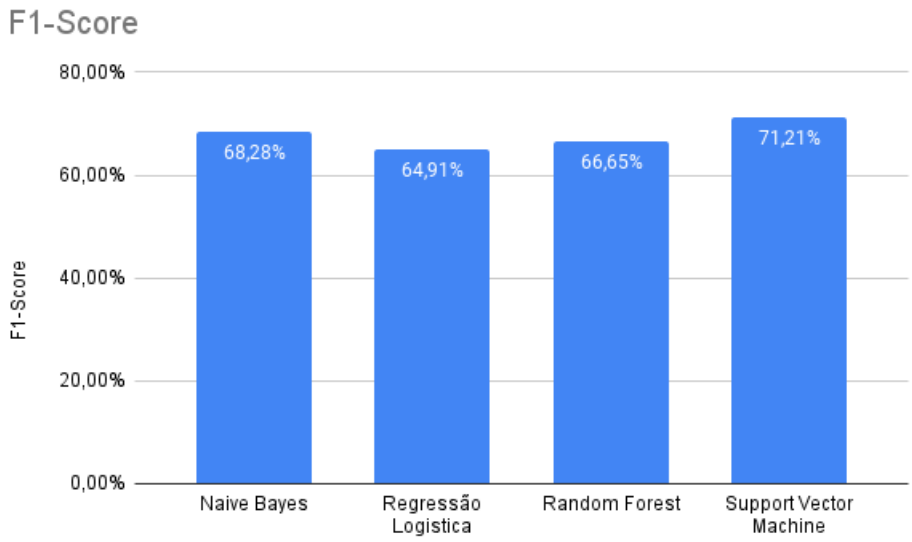
Fonte: Autor (2021)

4.5 RESULTADOS AGRUPADOS

Visando facilitar a comparação entre os modelos, na Tabela 14 estão sumarizados todos os resultados dos algoritmos já anteriormente apresentados individualmente.

Na Figura 11, está demonstrado um gráfico da métrica de F1-Score dos modelos analisados, onde podemos concluir que o melhor modelo é o de *Support Vector Machine*, seguido do *Naïve Bayes*, *Random Forest* e *Regressão Logística*.

Figura 11 – Gráfico comparativo dos modelos.



Fonte: Autor (2021)

Tabela 14 – Resultados Agrupados

Classe	Métrica	*Naïve Bayes*	*Regressão Logística*	*Random Forest*	*SVM*
Micro Onibus	Recall	97,16%	99,66%	100,00%	99,92%
	Precision	99,89%	99,62%	99,99%	99,98%
	F1-Score	98,51%	99,64%	99,99%	99,95%
	Accuracy	97,26%	99,33%	99,99%	99,92%
	Balanced Accuracy	97,89%	97,24%	99,97%	99,86%
Midi Onibus	Recall	98,24%	94,82%	99,95%	99,80%
	Precision	71,71%	95,35%	99,95%	98,99%
	F1-Score	82,91%	95,08%	99,95%	99,39%
	Accuracy	97,23%	99,33%	99,99%	99,91%
	Balanced Accuracy	97,70%	97,24%	99,97%	99,86%
Padron	Recall	100,00%	0,00%	0,00%	7,69%
	Precision	13,26%	0,00%	0,00%	100%
	F1-Score	23,42%	0,00%	0,00%	14,28%
	Accuracy	99,97%	99,99%	99,99%	99,99%
	Balanced Accuracy	99,98%	49,99%	50,00%	53,84%
Médias	Recall	98,47%	64,83%	66,65%	69,14%
	Precision	61,62%	64,99%	66,65%	99,66%
	F1-Score	68,28%	64,91%	66,65%	71,21%
	Accuracy	98,15%	99,55%	99,99%	99,94%
	Balanced Accuracy	98,52%	81,49%	83,31%	84,52%

Fonte: Autor (2021)

5 CONCLUSÃO

Este trabalho demonstrou a aplicação de modelos de aprendizado de máquina para recomendar veículos de transporte coletivo. Inicialmente foi apresentado o processo de extração, transformação e limpeza dos dados de entrada para os modelos de aprendizado de máquina selecionados. Na sequência foram realizados experimentos para avaliar o desempenho dos modelos. Considerando a métrica F1-Score o algoritmo SVM demonstrou os melhores resultados, seguido de do algoritmo Naïve Bayes, Random Forest e Regressão logística.

O trabalho demonstrou um grande potencial de utilização na rotina operacional da empresa de transporte coletivo. Com o modelo de previsão gerado, atingindo o objetivo do trabalho que era a geração de um modelo e indicação de categorias de veículos baseado na demanda.

5.1 TRABALHOS FUTUROS

No decorrer deste trabalho notou-se algumas possibilidades de trabalhos futuros, sendo elas a criação de um *hardware* para a contagem de desembarque de passageiros, incrementando a precisão dos dados de contagem de passageiros e dispensando a adoção do limiar adotado neste trabalho.

Referências

BONESSO, D. Estimação dos Parâmetros do Kernel em um Classificador SVM na Classificação de Imagens Hiperespectrais em uma Abordagem Multiclasse. 2013. Citado na página 17.

CABRAL, C. I. S. Aplicação do Modelo de Regressão Logística num Estudo de Mercado. **Dissertação de Mestrado-sboa, Faculdade de Ciências. (Departamento de Estatística e Investigação Operacional)cias.**, p. 1–59, 2013. Citado na página 19.

FORSYTH, D. **Probability and Statistics for Computer Science**. 1st. ed. [S.l.]: Springer, 2018. ISBN 9783319644097. Citado na página 17.

GOLDSCHMIDT, R.; PASSOS, E.; BEZERRA, E. **Data Mining: Conceitos, técnicas, algoritmos e aplicações**. 2º. ed. [S.l.]: Elsevier Editora LTDA., 2015. ISBN 9788535278224. Citado na página 16.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning Data Mining, Inference, and Prediction**. 2º. ed. [S.l.: s.n.], 2001. ISBN 9780387848570. Citado na página 17.

HOSMER, D. W.; LEMESHOW, S.; STURDIVANT, R. X. **Applied Logistic Regression**. John Wiley & Sons, Inc., 2013. Disponível em: <https://doi.org/10.1002/9781118548387>. Citado na página 19.

MELLO, R. F. de; PONTI, M. A. **Machine Learning - A Practical Approach on the Statistical Learning Theory**. [S.l.: s.n.], 2018. 1–373 p. ISSN 01697161. ISBN 9781405164535. Citado 2 vezes nas páginas 16 e 17.

PROVOST, F.; FAWCETT, T. **Data Science for Business: What You Need to Know About Data Mining and Data-analytic Thinking**. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2013. ISBN 1449361323, 9781449361327. Citado 4 vezes nas páginas 14, 18, 20 e 23.

RUSSELL, S. J.; NORVIG, P.; DAVIS, E. **Artificial intelligence : a modern approach**. 3º. ed. [S.l.: s.n.], 2013. 1132 p. ISBN 9780136042594. Citado 2 vezes nas páginas 14 e 16.

SIMON, P. **Too Big to Ignore: The Business Case for Big Data**. 1st. ed. [S.l.]: John Wile & Sons, 2013. ISBN 781118638170, 9781118642108, 9781118641682, 9781118641866. Citado na página 14.

SKLEARN. **SKLearn Calibration Plot**. 2015. Disponível em: <https://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html#sphx-glr-auto-examples-calibration-plot-compare-calibration-py>. Acesso em: 22 de maio de 2020. Citado na página 22.

TUFFÉRY, S. **Data Mining and Statistics for Decision Making**. [S.l.: s.n.], 2011. ISBN 9780470688298. Citado na página 21.

VUK, M.; CURK, T. ROC curve, lift chart and calibration plot. **Metodološki zvezki**, v. 1, n. 3, p. 89–108, 2006. ISSN 1854-0023. Citado na página 21.

WIKIMEDIA. **Lift curve**. 2006. Disponível em: <https://commons.wikimedia.org/wiki/File:Lift_curve.svg>. Acesso em: 22 de maio de 2020. Citado na página 21.

WIKIMEDIA. **Wikimedia Regression Logistic**. 2008. Disponível em: <https://commons.wikimedia.org/wiki/File:Logistic-curve.svg>. Acesso em: 27 de maio de 2020. Citado na página 19.

WIKIMEDIA. **Wikimedia ROC**. 2009. Disponível em: <https://commons.wikimedia.org/w/index.php?curid=8326140>. Acesso em: 24 de abril de 2020. Citado na página 20.

WIKIMEDIA. **Wikimedia SVM**. 2017. Disponível em: <https://commons.wikimedia.org/wiki/File:Nonlinear_SVM_example_illustration.svg>. Acesso em: 22 de abril de 2020. Citado na página 18.

WIKIMEDIA. **Wikimedia Random Forest**. 2020. Disponível em: <https://commons.wikimedia.org/wiki/File:Random_forest_diagram_complete.png>. Acesso em: 25 de abril de 2020. Citado na página 19.

WIKIPEDIA. **Wikipédia**. 2020. Disponível em: <https://en.wikipedia.org/wiki/Support-vector_machine#/media/File:SVM_margin.png>. Acesso em: 22 de abril de 2020. Citado na página 18.

XUE, R.; SUN, D. J.; CHEN, S. Short-term bus passenger demand prediction based on time series model and interactive multiple model approach. **Discrete Dynamics in Nature and Society**, v. 2015, 04 2015. Citado na página 15.