LETICIA MANCUZO DE ALMEIDA
WILLIAN WERNER ANGELO DA COSTA

INFERÊNCIA DE INDICADORES SOCIO-ECONÔMICOS COM BASE EM IMAGENS DE RUA UTLIZANDO *DEEP LEARNING*

São Paulo
2020

LETICIA MANCUZO DE ALMEIDA
WILLIAN WERNER ANGELO DA COSTA

INFERÊNCIA DE INDICADORES SOCIO-ECONÔMICOS COM BASE EM IMAGENS DE RUA UTLIZANDO *DEEP LEARNING*

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção dos Títulos, respectivamente, de Engenheira de Controle e Engenheiro de Computação.

São Paulo
2020

LETICIA MANCUZO DE ALMEIDA
WILLIAN WERNER ANGELO DA COSTA

INFERÊNCIA DE INDICADORES SOCIO-ECONÔMICOS COM BASE EM IMAGENS DE RUA UTLIZANDO *DEEP LEARNING*

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção dos Títulos, respectivamente, de Engenheira de Controle e Engenheiro de Computação.

Área de Concentração:

Engenharia de Controle e Engenharia de Computação

Orientador:

Pedro Luiz Pizzigatti Corrêa

Co-orientador:

Marina Jeaneth Machicao Justo

São Paulo
2020

RESUMO

Utilizando imagens de rua extraídas do *Google Street View* e dados demográficos do Censo IBGE 2010, esse estudo propõe um modelo preditivo baseado em *deep learning* para ser usado como estimador de indicadores socio-econômicos. Além disso, apresenta-se os experimentos intermediários e os ajustes que foram feitos à metodologia do experimento final.

Palavras-Chave – *deep Learning*, *Big Data*, emphData Science.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

PARTE I

INTRODUÇÃO

1 DESCRIÇÃO

Pretende-se fazer um modelo que consiga predizer indicadores socioeconômicos e demográficos, usando como entrada imagens das ruas de cada região analisada. Para tal, utiliza-se um aprendizado supervisionado de máquina a partir de dados censitários com a técnica de *deep learning*. A metodologia do projeto foi baseada em ciclos de *Data Science*, de forma que para cada ciclo repetimos os processos presentes num projeto de big data. (Ver apêndice A)

1.1 Motivação

Atualmente no Brasil indicadores socioeconômicos são extraídos de pesquisas de censo realizadas pelos governos com periodicidade de 10 anos. Tais pesquisas são feitas através de questionários aplicados presencialmente em todas as casas do país, tornando-as muito caras, demoradas e inviáveis de serem realizadas com uma frequência maior. Além de estarem sujeitas a imprevistos, como o ocorrido com a pandemia do coronavírus no ano de 2020, que adiou a data de realização do censo. Tendo isso em vista, e que para poder realizar um acompanhamento minucioso do progresso das metas de desenvolvimento social, econômico e ambiental do país deve-se ter uma atualização mais frequentes dos dados. Portanto é importante explorar formas alternativas para a obtenção desses dados. A aplicação de *Big Data* surge como uma boa alternativa para esse problema, dado o processamento massivo de informações necessário.

2 OBJETIVO

Este projeto tem como objetivo desenvolver um sistema de Big Data capaz de obter imagens do *Google Street View* do Vale do Ribeira, processá-las e, com base em múltiplos modelos de domínio específicos, extrair *features* capazes de determinar para cada município seus indicadores socio-econômicos. Tais dados correspondem aos extraídos nos censos demográficos do IBGE, os quais serão posteriormente utilizados como base de treino para os modelos de regressão.

3 REVISÃO DA LITERATURA

Neste capítulo foi feito o levantamento de vários artigos recentes na literatura que abordam os assuntos de *deep learning*, imagens de rua, imagens de satélites como preditores de pobreza.

A primeira seção desse capítulo diz sobre o artigo aqui replicado no experimento inicial; os demais são artigos relevantes e relacionados, que serviram de base para todos os experimentos apresentados.

3.1 Suel et Al (2019). Measuring social, environmental and health inequalities using deep learning and street imagery [1]

Nesse artigo mostra-se a criação de um modelo de *deep learning* que, a partir de imagens de rua do *Google Street View* de diversas regiões de Londres e outras cidades da Inglaterra, classifica a distribuição espacial dessas regiões para diversas estatísticas socioeconômicas, tais como renda, educação, desemprego, moradia, ambiente de vida, saúde e criminalidade. Utiliza-se de *transfer learning* utilizando como base o VGG16, um modelo de detecção de objetos gerais em imagens para gerar um vetor de representação, que contém informações relevantes, embora latentes, sobre as imagens.

O artigo exibe diversas métricas de performance, apresentando bons resultados, e conclui mostrando o potencial que as imagens possuem de complementar as pesquisas tradicionais de dados socioeconômicos e monitorar políticas públicas relacionadas. Na figura 1 temos exemplos de predições feitas pelo modelo e as imagens utilizadas.

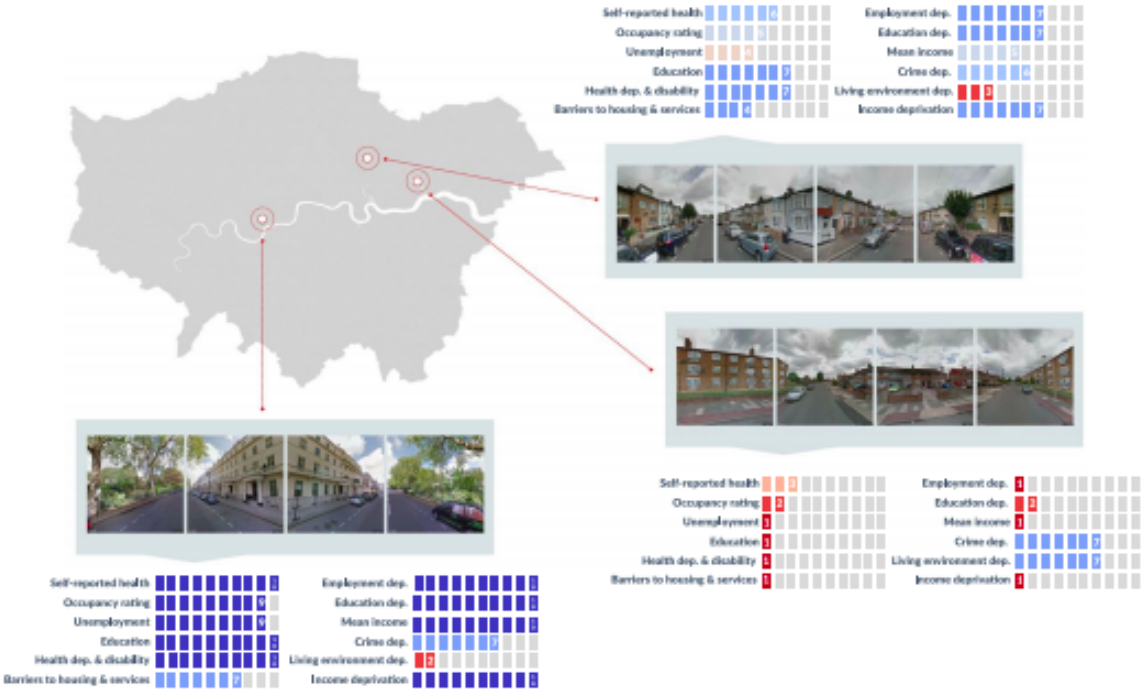


Figura 1: Ilustração de Suel et Al (2019) [1]

3.2 Jean et Al (2016). Combining satellite imagery and machine learning to predict poverty [2]

Aqui parte-se de uma abordagem muito conhecida para estimar variáveis socioeconômicas relacionadas à pobreza: o uso de imagens de satélite noturnas de alta resolução. A abordagem consiste no uso de *transfer learning* com imagens do *Google Static Maps* para predizer a intensidade de luz noturna e para predizer as variáveis socioeconômicas.

O artigo usa os países da áfrica como região de escopo e tem como menor subdivisão os *clusters*, que são aproximadamente os vilarejos, do qual se extraem dados de consumo diário doméstico e de renda.

Na figura 2 temos ilustrado alguns resultados do modelo.

3.3 LeCun et Al (2015). Deep learning [18]

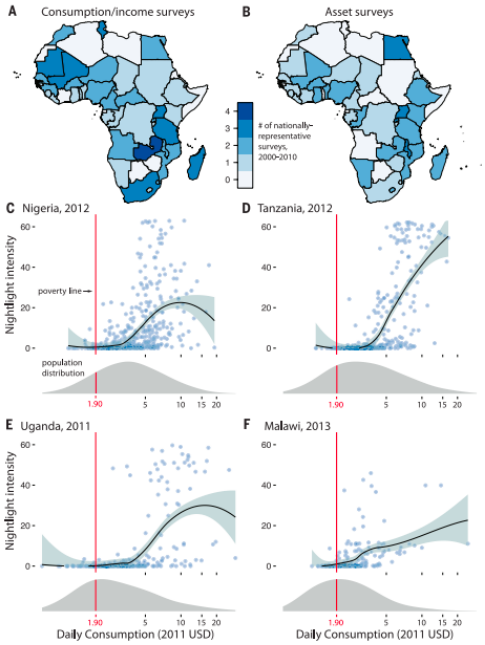


Figura 2: Ilustração de Jean et Al (2016) [2]

4 ASPECTOS CONCEITUAIS

4.1 Machine Learning

O *Machine Learning* é a ciência de programar computadores para que eles possam aprender através de dados. Tom Mitchell e Aurélien Géron, 1997, definiram como "Um programa de computador que tem a responsabilidade de aprender sobre experiência E a respeito de uma tarefa T com uma performance P, se sua performance em T, como medido por P melhorar a experiência E" (Tom Mitchell apud Aurélien Géron, 1997).

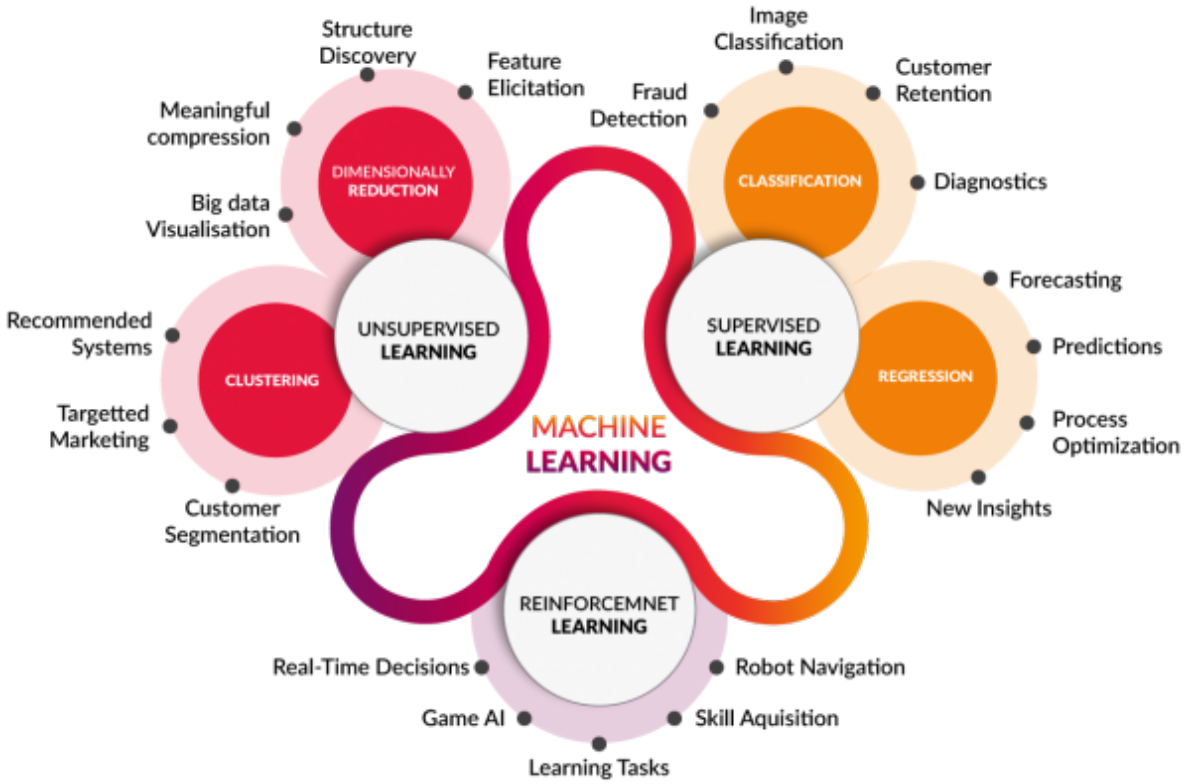


Figura 3: Tipos de Aprendizado de Máquina [3]

E essa análise de dados em larga escala torna o *Machine Learning* uma importante ferramenta de extrair informações a partir dos dados e inferir previsões de um evento, utili-

zando conceitos de Inteligência Artificial, Ciência da Computação e Análises Estatísticas.

O grande diferencial da técnica de aprendizado de máquina em relação à programação usual é a não necessidade de se codificar todas as possibilidades existentes do problema que gera especificação de domínio e grande conhecimento do modelo de negócio. Dessa forma, os algoritmos de *Machine Learning* não precisam de blocos lógicos explícitos, na verdade, eles levam em consideração análises estatísticas associando entradas e saídas de dados e suas possíveis correlações.

4.1.1 Tipos de Aprendizado

Os tipos de aprendizagem em aprendizado de máquina são divididos em três categorias.

4.1.1.1 Aprendizado Supervisionado

Aprendizado supervisionado é o tipo de aprendizagem mais comum de *machine learning* atualmente.

No modelo de aprendizado supervisionado, os dados de treinamento de um algoritmo possuem a solução desejada para o problema, portanto, o programa já sabe qual saída/resultado deve ser considerado como correto (denominado *label*), dada a entrada de dados (denominadas *features*). Após o aprendizado da máquina com os exemplos dados, é possível utilizar esse modelo treinado para fazer predições de novas instâncias de dados. Os algoritmos supervisionados podem ser subdivididos em algoritmos de classificação e algoritmos de regressão. Os algoritmos de classificação têm como objetivo classificar itens ou amostras de acordo com as características observadas pelo supervisor. Enquanto os algoritmos de regressão funcionam com a compreensão de relação da máquina, quanto às variáveis para prever valores. Ou seja, em classificações a nossa variável resposta deve ser discreta, enquanto em regressões esta pode ser contínua.

4.1.1.2 Aprendizado Não Supervisionado

Como pode-se deduzir, o aprendizado não supervisionado é igual ao aprendizado supervisionado, porém sem a utilização de *labels*. Dessa forma, o sistema tenta aprender sem ter os exemplos de classificação já corretos e definidos. Esse tipo de aprendizagem é comumente utilizado quando se tem muitos dados e é necessário agrupá-los pelas características em comum, ou seja, *clusterizar* as amostras; ou para determinar padrões escondidos que

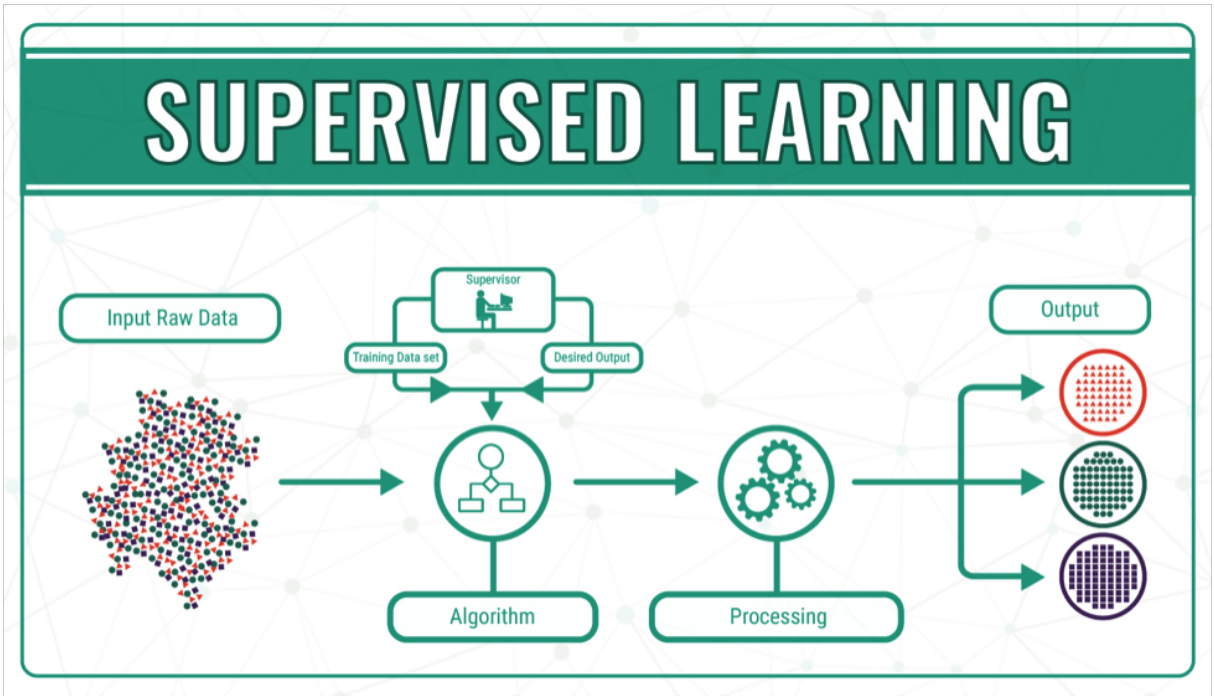


Figura 4: Aprendizado Supervisionado [4]

não são de fáceis reconhecimento. Normalmente os dados possuem muitas dimensões, o que dificulta bastante a visualização organizada, por meio de gráficos e mapas e, então, o aprendizado não supervisionado vem para determinar como esses dados estão distribuídos no espaço, isto é, a estimativa de densidade. Porém um grande problema do aprendizado não supervisionado é que não é possível testar e validar os resultados após a construção do modelo. Assim, muitas vezes o aprendizado não supervisionado é utilizado em conjunto com o aprendizado supervisionado.

O aprendizado não supervisionado pode ser dividido em duas categorias:

Aprendizagem Paramétrica Não Supervisionada

Nesse caso, assume-se uma distribuição paramétrica de dados. O algoritmo assume que os dados da amostra vêm de uma população que segue uma distribuição de probabilidade com base em um conjunto fixo de parâmetros. Teoricamente, em uma família normal de distribuições, todos os membros têm a mesma forma e são parametrizados por média e desvio padrão. Ou seja, conhecendo a média e o desvio padrão e se a distribuição é normal, conhece-se a probabilidade de qualquer observação futura. A aprendizagem paramétrica não supervisionada envolve a construção de modelos de mistura gaussiana e o uso do algoritmo de maximização da expectativa para prever a classe da amostra em questão.

Aprendizagem Não Supervisionada Não Paramétrica

Na versão não parametrizada do aprendizado não supervisionado, os dados são agrupados em *clusters*, onde cada *cluster* (esperançosamente) diz algo sobre categorias e classes presentes nos dados. Este método é comumente usado para modelar e analisar dados com pequenos tamanhos de amostra. Ao contrário dos modelos paramétricos, os modelos não paramétricos não exigem que o modelador faça suposições sobre a distribuição da população e, portanto, às vezes são chamados de método livre de distribuição.

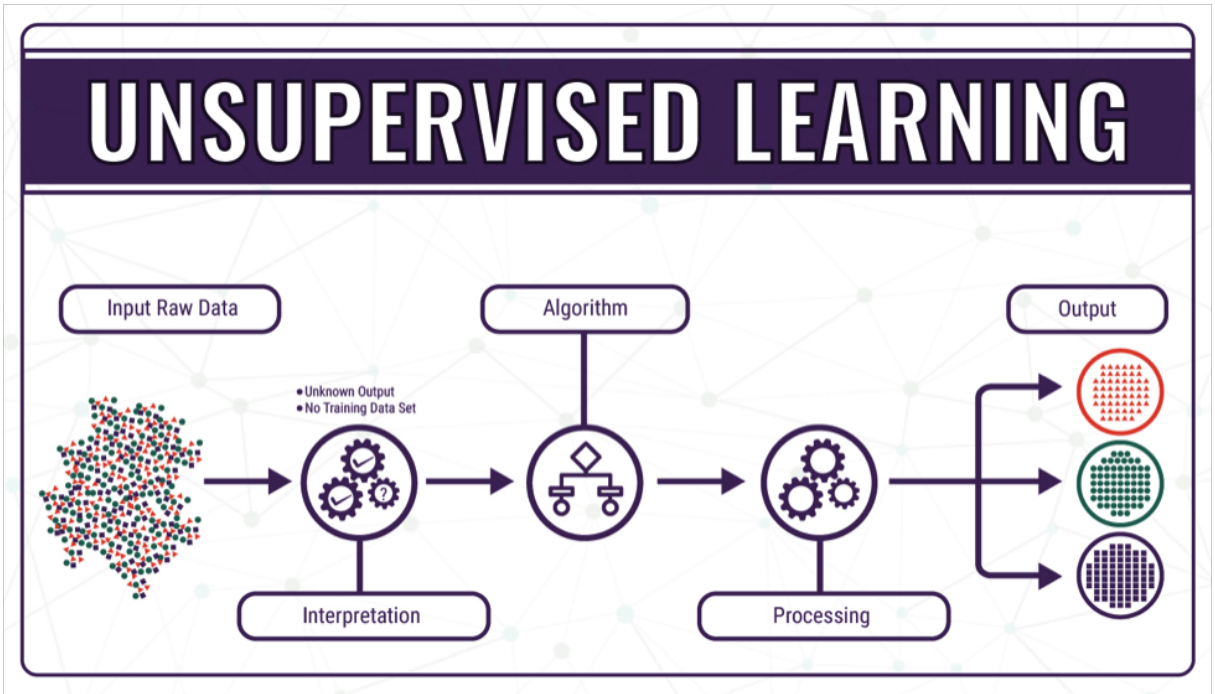


Figura 5: Aprendizado Não Supervisionado [5]

4.1.1.3 Aprendizado por Reforço

O aprendizado por reforço é o que mais difere dos mencionados anteriormente, pois é um sistema que possui um agente capaz de observar o ambiente e tomar ações, recebendo recompensas ou penalidades como retorno dessa ação. De maneira autônoma ele deve gerar uma política que mais trará recompensas do que penalidades ao longo prazo. Essa política, então, define qual a ação deve ser tomada dado o ambiente atual.

Todo algoritmo de *Deep Reinforcement Learning* segue um processo chamado Processo de Decisão de Markov (MDP), que é uma tupla com cinco valores:

1. **Conjunto Finito de Estados (S)**: Cada projeto terá seu próprio conjunto de estados que não mais é que todas as possibilidades de estado do projeto. Por exemplo: andar, pular e esperar.

2. **Conjunto Finito de Ações**: Ações que o modelo executa, ou seja, as ações de transições de cada estado.

3. **Modelo de probabilidade (P)**: Probabilidade de passar de um estado para outro.

4. **Recompensa (R)**: A Recompensa é um valor número que o modelo irá receber após realizar aquela Ação naquele Estado. Quando esse número é positivo, quer dizer que a Ação feita naquele Estado foi benéfica; caso o número seja negativo, o modelo será punido por aquela Ação. O objetivo principal do modelo é realizar o conjunto de Ações que maximizam a Recompensa até seu objetivo final.

5. **Fator de Desconto (Y)**: O fator de desconto é um número escalar, geralmente entre 0 e 1. Ele influencia no total de recompensa futura que o seu Agente irá receber. Em outras palavras, o modelo irá preferir percorrer caminhos na fase que irão garantir uma recompensa maior no futuro, ao invés de uma Recompensa imediata.

Com base nessas cinco propriedades base, um algoritmo de *reinforcement learning* é capaz de aprender, através de tentativa e erro, quais Ações (A) ele deve fazer em cada Estado (S) a fim de que sua Recompensa (R) seja a maior possível para atingir um Objetivo. Durante o treinamento, o Agente irá tentar todas as possibilidades e selecionar a que garantir um melhor resultado no final.

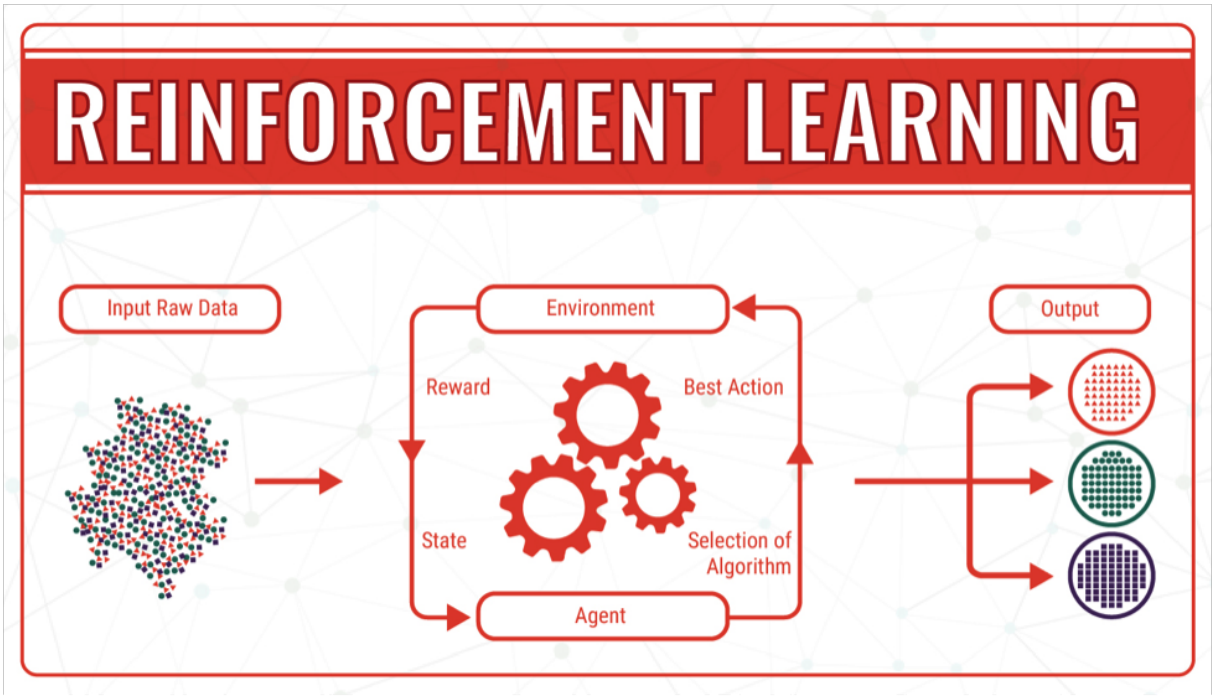


Figura 6: Aprendizado por Reforço [6]

4.1.1.4 *Overfitting* e *Underfitting*

No caso do aprendizado supervisionado, através dos dados de treinamento a máquina realiza o ajuste do modelo, o *model fit*. Porém quando esses dados não são suficientes, ou estão desbalanceados (uma classe possui bem mais quantidade do que outra) a máquina acaba apresentando dificuldades em ajustar o modelo, ocorrendo *Overfitting* ou *Underfitting*. O *Oberfitting* é quando o modelo consegue prever muito bem os dados de treinamento, porém para os dados de teste essa performance cai consideravelmente. Ou seja, o modelo não consegue generalizar muito bem os casos. Esse tipo de erro pode ocorrer tanto pelo algoritmo em si, mas em geral ocorre pela qualidade dos dados ou pela falta deles, isso porque um *dataset* muito pequeno ou com muito ruído (dados que não possuem relevância nenhuma no problema) leva o modelo a encontrar padrões que funcionam bem para os dados de teste, porém quando são levados novos exemplos o modelo não conseguirá acertar. Para resolver esse problema é necessário fazer regularizações, isto é, leves ajustes que permitem o modelo ter meior grau de liberdade e se ajustar melhor sobre os dados de treinamento. Essas regularizações podem ser controladas por hiperparâmetros, que são parâmetros do algoritmo de aprendizado e não do modelo em si. O ajuste desses hiperparâmetros é um importante passo para a construção de um modelo de *Machine Learning*, e identificar um valor adequado para esses parâmetros é um grande desafio, visto que uma configuração ruim pode gerar o *overfitting* ou o *underfitting*. Caso sejam configurados números muito pequenos o risco do modelo continuar na situação de *overfitting* se eleva. Já no caso de se colocar valores muito altos para os hiperparâmetros, o modelo ficará muito linear e simplificado, o que, por um lado, eliminará de vez o risco de *overfitting* porém aumentará a chance de ocorrer *underfitting*. O *Underfitting*, como é possível deduzir, é o contrário do *overfitting*, ou seja, ocorre quando o modelo final gerado é simples demais em comparação com a estrutura dos dados. O desafio, então, é realizar o ajuste fino dos hiperparâmetros de forma a adequar o modelo dentro da estrutura de dados, evitando tanto o *overfitting* quanto o *underfitting*, aumentando a sua performance em dados reais ou de teste.

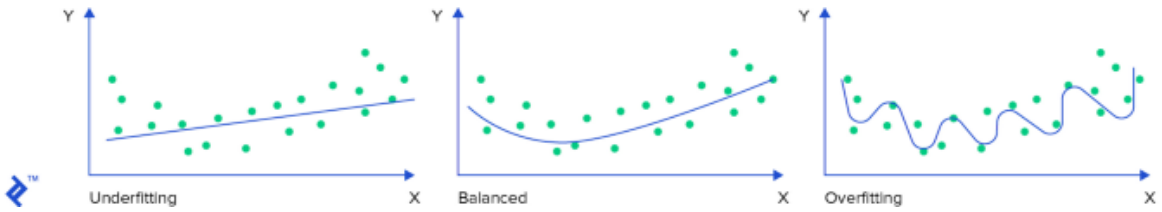


Figura 7: Ilustração *Overfitting* e *Underfitting* [7]

4.1.1.5 *Oversampling* e *Undersampling*

Os modelos de *machine learning* são construídos, entre outros objetivos, para minimizar erros. Porém em vários casos, quando estamos trabalhando com modelos de classificação, observamos que a probabilidade de uma amostra aleatória em nosso *dataset* pertencer a uma determinada classe é bem maior que a outra. E com isso o algoritmo estará enviesado para generalizar classificações posteriores. Assim, quando situações como essa ocorrem, na etapa de preparação dos dados precisamos usar certas técnicas, chamadas de *oversampling* e *undersampling*.

Oversampling

O *oversampling* é quando replicamos os casos da classe com menos casos, até chegarmos ao valor, ou próximo do valor, da classe com mais casos. Essa técnica é bastante perigosa, pois pode levar muito rápido o modelo ao *overfitting*. Assim recomenda-se usar técnicas mais avançadas de *overfitting*, como por exemplo:

- *Smote* : O primeiro passo dessa técnica é encontrar os vizinhos próximos para as classes em minoria para cada amostra das classificações. Em seguida, traça-se uma reta entre o ponto original e o vizinho para definir a localização da observação da observação genérica.

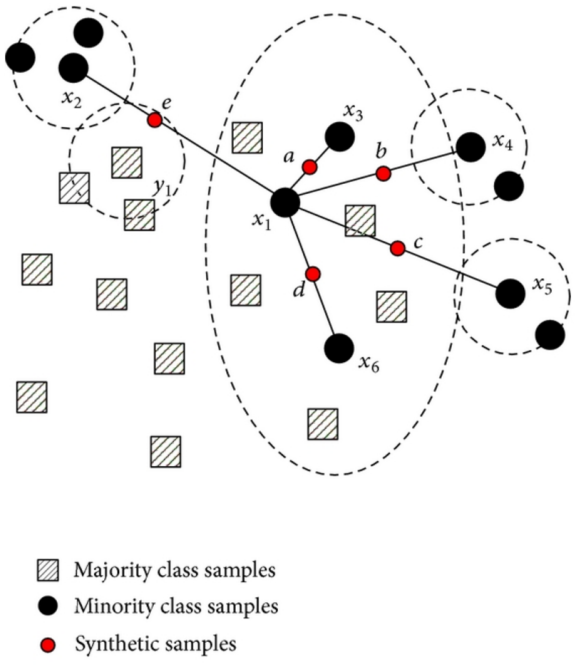


Figura 8: Ilustração Técnica *Smote* [8]

- *Adasyn* : Essa técnica é uma versão melhorada da técnica *smote*. Ela faz inicialmente tudo o que o *smote* faz porém depois de criar a amostra, ele adiciona valores aleatórios pequenos aos pontos, tornando-os mais realistas. Em outras palavras, ao invés de toda a amostra ser linearmente correlacionada a origem, como no *smote*, com o *adasyn* os valores possuem um pouco mais de variância entre eles, e com isso são mais dispersos.

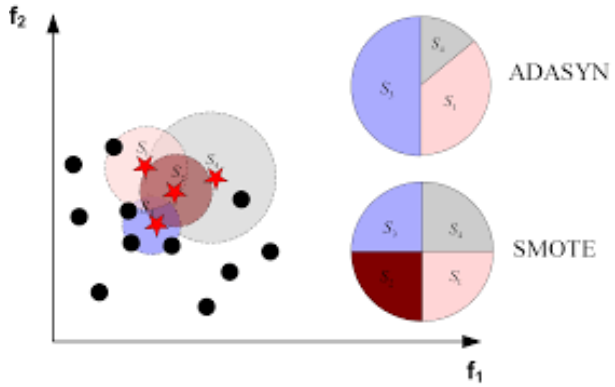


Figura 9: *Smote* x *Adasyn* [9]

Undersampling

Agora o *Undersampling* é quando reduzimos os casos da classe com mais casos até chegar no mesmo nível da classe com menos casos. Essa técnica é só recomendada quando há grande abundância de dados, pois quanto menos dados disponíveis a chance do modelo não se ajustar corretamente aumenta consideravelmente, podendo ocorrer problemas de *overfitting* ou *underfitting*. Assim, como no *oversampling* o ideal é usar técnicas mais avançadas, como por exemplo:

- *Neighbourhood Cleaning Rule*: Nessa técnica aplica-se o algoritmo dos vizinhos próximos (KNN *k-nearest neighbors algorithm*)e remove as observações que não se enquadram. A cada iteração aumenta-se a quantidade de vizinhos próximos no modelo.

- *AllKNN*:Essa técnica é bem parecida com a anterior, porém nessa o foco é limpar os dados e não apenas em condesálos.

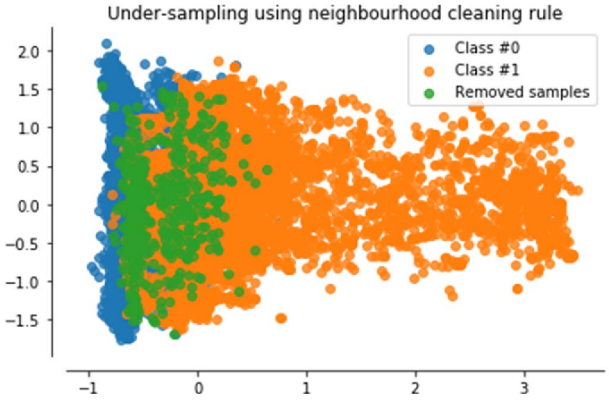


Figura 10: Esquema algoritmo *Neighbourhood Cleaning Rule [10]*

4.1.2 Modelos de Treinamento

4.1.2.1 Rede Neural Artificial

Redes Neurais Artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento, cujo funcionamento é bastante simples. Essas unidades, geralmente são conectadas por canais de comunicação que estão associados a determinado peso e passam por uma determinada função de ativação. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões. O comportamento inteligente de uma Rede Neural Artificial vem das interações entre as unidades de processamento da rede. As redes neurais podem ser bastante diferentes entre si, porém a sua estrutura básica segue como na figura 11. O que vai diferenciar as redes neurais entre si e deixá-las mais complexas são os parâmetros e hiperparâmetros. Os parâmetros são coeficientes do modelos, e são escolhidos pelo próprio modelo. Isso quer dizer que o algoritmo, enquanto está aprendendo, otimiza certos coeficientes e escolhe os parâmetros que minimizarão o erro. Assim a parte de escolha de parâmetros dependerá do modelo que está sendo utlizado e não com escolhas do programador de fato. Já os hiperparâmetros, em contra-ponto, precisam ser escolhidos.

Parâmetros Como já mencionado, quem irá determinar os parâmetros finais do algoritmo é o próprio modelo, porém deve-se inicializar esses valores. Para uma inicialização correta e que não irá atrapalhar posteriormente o algoritmo deve-se utilizar uma inicialização com distribuição normal ou uniforme. A melhor irá depender de qual função de ativação escolhida nos hiperparâmetros.

Hiperparâmetros

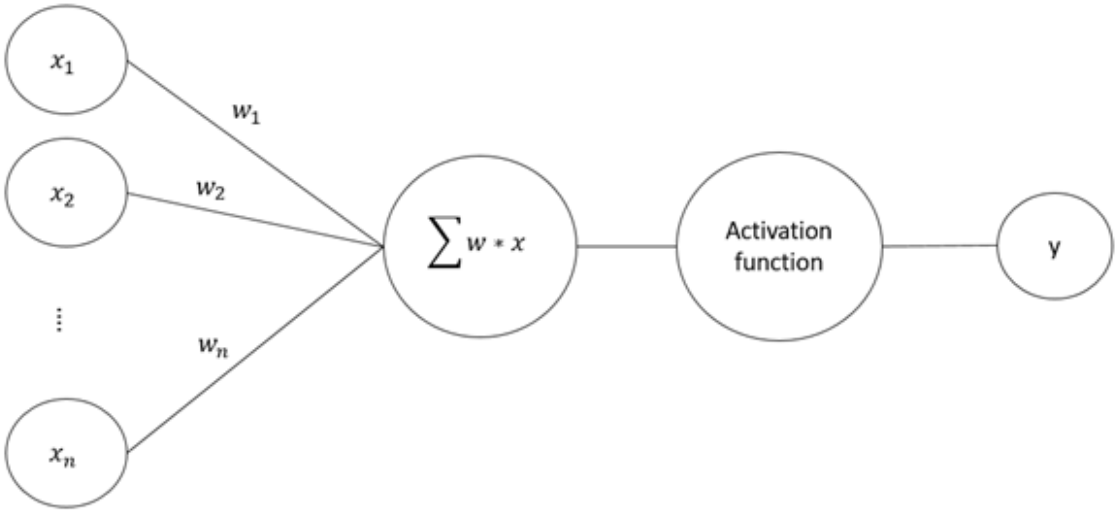


Figura 11: "Esqueleto"Rede Neural Artificial [11]

- **Número de Camadas Escondidas**: A melhor abordagem para escolher o total de camadas escondidas em um rede é a manual, por tentativa e erro. Quanto menos camadas a rede tiver, mais rápido será o processamento e mais generalizada ela vai ser. Porém, ela não pode ter camadas de menos, já que assim vai generalizar demais, acontecendo o *underfitting*.

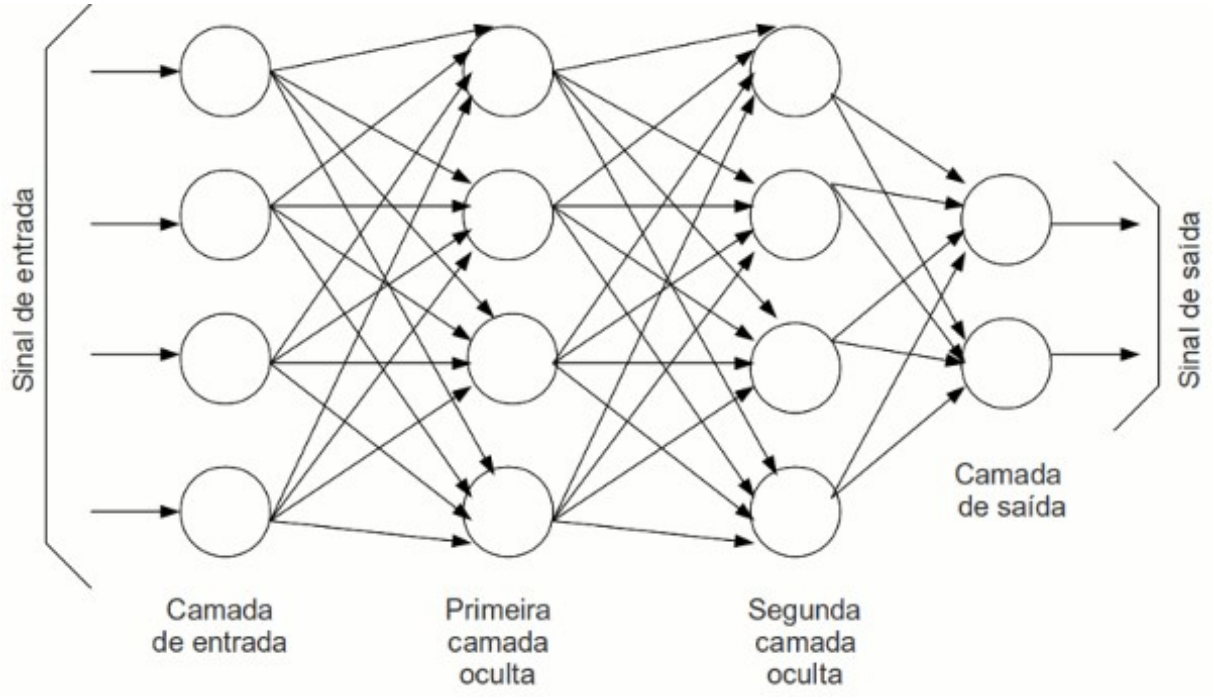


Figura 12: Esquema Rede Neural Artificial [12]

- **Taxa de Aprendizagem**: É a taxa com a qual o algoritmo irá aprender, ou seja, quão rápido ele chegará no ponto de mínimo da função custo. O problema de uma taxa pequena é que a rede convergirá bem lentamente e podemos cair no caso do *"Vanishing Gradiente"*, isto é, quando a rede nunca chega ao ponto de mínimo. Porém ao escolher uma taxa de aprendizagem muito grande, a chance de não encontrar o ponto de mínimo aumenta bastante. Assim, o ideal é uma taxa de aprendizagem não tão baixa e não tão alta.

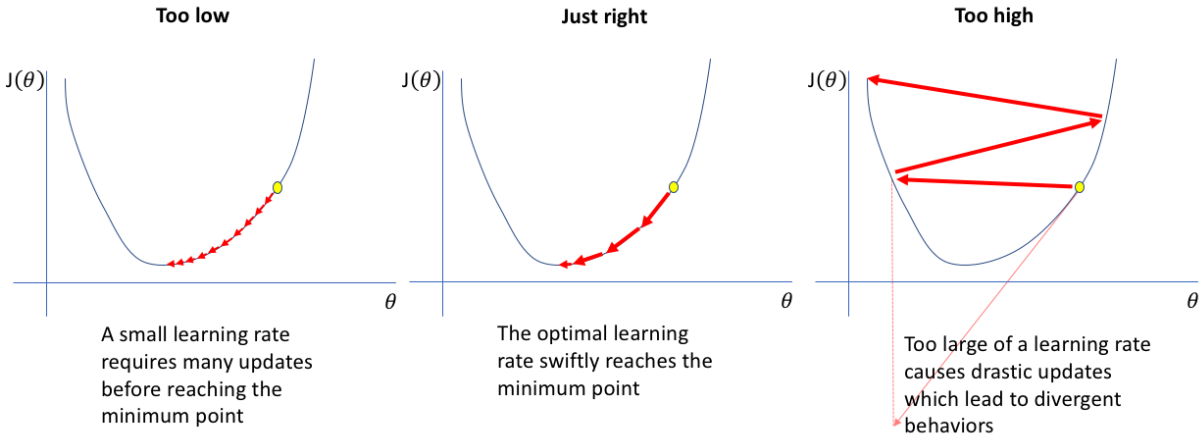


Figura 13: Exemplos taxa de Aprendizagem [13]

- **Momento**: É a técnica usada durante a fase do algortimo *backpropagation*.Como dito em relação à taxa de aprendizado, os parâmetros são atualizados para que possam convergir para o mínimo da função custo. Porém esse processo pode acabar ficando muito demorado e afetar a eficiência do algoritmo. Portanto, uma solução possível é acompanhar as direções anteriores (que são os gradientes da função custo em relação aos pesos) e mantê-los como informações embutidas, o momento. Com ele, aumenta-se a velocidade de convergência não em termos de taxa de aprendizagem (quanto um peso é atualizado a cada vez), mas em termos de memória embutida de recalibração anterior (o algoritmo sabe que a direção anterior desse peso estava, digamos, correta , e continuará diretamente nessa direção durante a próxima propagação).

- **Função de Ativação**: É a função pela qual passa-se a soma ponderada, a fim de ter uma saída significativa, ou seja, como um vetor de probabilidade ou uma saída 0-1. As principais funções de ativação são Sigmoid (para classificação multiclasse, uma variante desta função é usada, chamada função SoftMax: ela retorna como saída um vetor de probabilidade cuja soma é igual a um), Tanh e RELU.

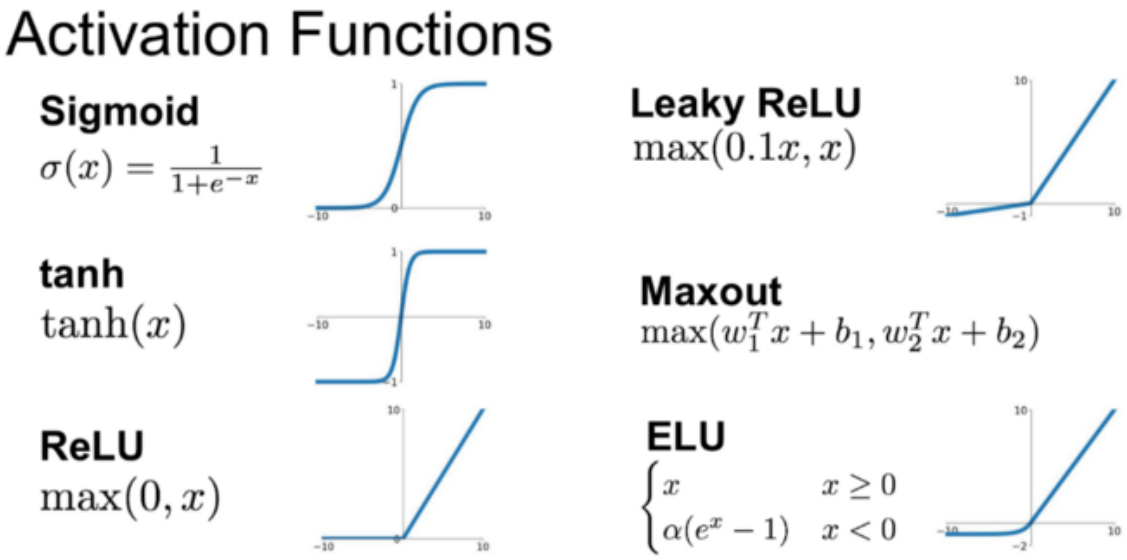


Figura 14: Tipos de Função de Ativação [14]

- ***Batch***: Quando o *dataset* utilizado para treinar a rede contém muitos dados, pode resultar ineficiente alimentar sua rede com todos eles. Uma boa prática é alimentá-lo com amostras menores de seus dados, chamadas de *batch*, pois fazendo isso, toda vez que o algoritmo treinar, ele treinará em uma amostra de mesmo tamanho.
- **Épocas**: É o total de vezes que o algoritmo irá treinar com o *dataset*

Algoritmo *Backpropagation*

O algoritmo *backpropagation* é como as redes neurais artificias calculam o gradiente da função de custo. Ele é o algoritmo-chave que faz o treinamento de modelos profundos algo computacionalmente tratável. Para as redes neurais modernas, ele pode tornar o treinamento com gradiente descendente até dez milhões de vezes mais rápido, em relação a uma implementação ingênua. Essa é a diferença entre um modelo que leva algumas horas ou dias para treinar e e outro que poderia levar anos. Além de seu uso em *Deep Learning*, o *backpropagation* é uma poderosa ferramenta computacional em muitas outras áreas, desde previsão do tempo até a análise da estabilidade numérica. Fundamentalmente, o *backpropagation* é uma técnica para calcular derivadas rapidamente, utilizando o conceito de regra da cadeia. Simplificando, após um ciclo inicial para frente entre as camadas da rede, o *backpropagation* realiza vários ciclos para trás (da camada de saída para a camada de entrada) enquanto ajusta os parâmetros do modelo (pesos e tendências).

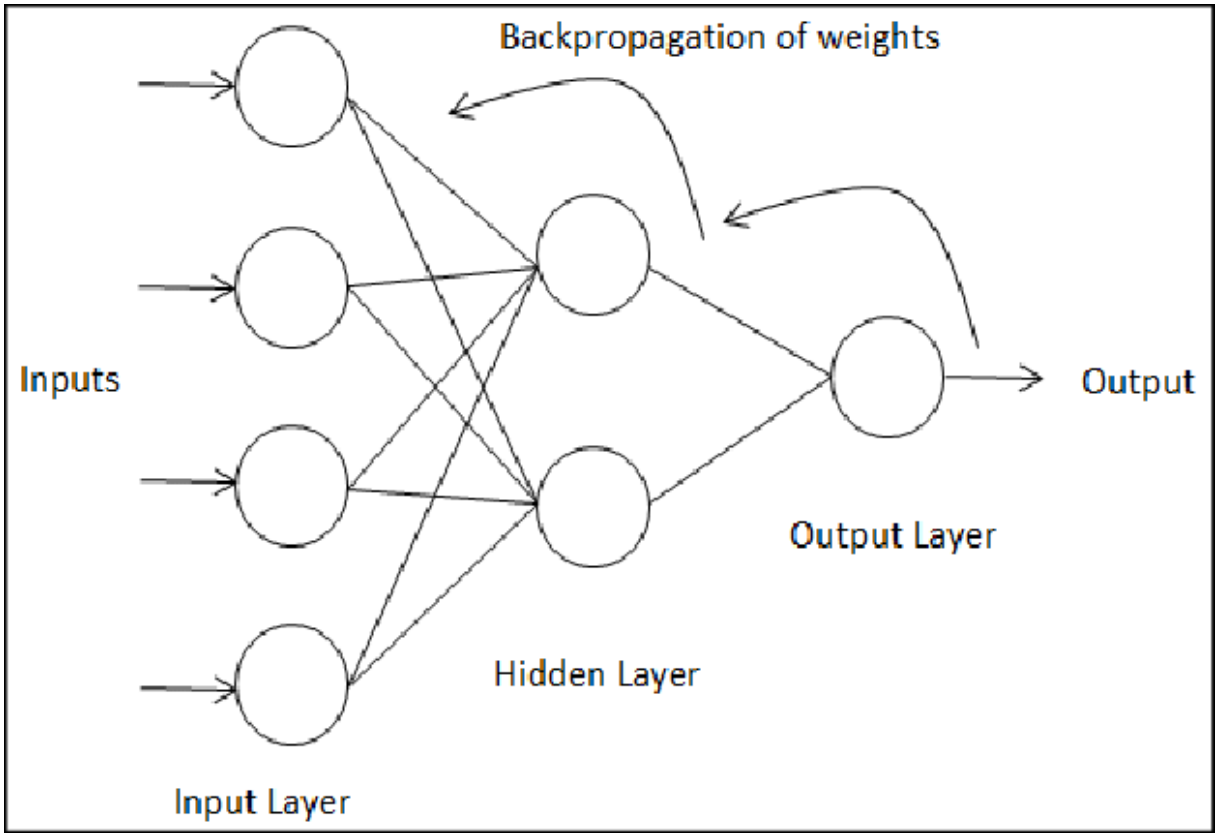


Figura 15: Esquema algoritmo *Backpropagation* [15]

4.1.2.2 *Deep Learning* (Aprendizagem Profunda)

Em [18], Bengio Hinton e LeCun ,ganhadores do prêmio Alan Turing em 2018, aprofundam e explicam os conceitos mais importantes sobre deep learning.

Deep learning é um tipo de *machine learning* que treina computadores para realizar tarefas como seres humanos, o que inclui reconhecimento de fala, identificação de imagem e previsões. Ao invés de organizar os dados para serem executados através de equações predefinidas, o *deep learning* configura parâmetros básicos sobre os dados e treina o computador para aprender sozinho através do reconhecimento padrões em várias camadas de processamento.

O método de *deep learning* permite os modelos computacionais a processarem várias camadas para conseguirem representar os dados com vários níveis de abstração. Esses métodos já ajudaram a melhorar drasticamente o reconhecimento de discursos e de objetos e em áreas da medicina e da genética. Além disso, a técnica de *deep learning*, aliada a técnica *backpropagation* que indica como a máquina deve mudar seus parâmetros internos para obter uma representação da camada anterior na camada seguinte, conseguiu descobrir estruturas complexas em largos *datasets* de dados.

As técnicas de aprendizado de máquina convencionais eram limitadas na capacidade de extrair informações da forma bruta dos dados, pois elas exigiam uma engenharia bem cuidadosa e uma considerável expertise. Porém com o surgimento do Aprendizado Representativo, do qual a aprendizagem profunda é um exemplo que utiliza diversas camadas inteligentes para concebê-lo, surgiu a possibilidade de, através da alimentação dos dados ao sistema, obter os parâmetros relevantes para a aplicação sem a necessidade de um especialista por trás da seleção. Em geral tais métodos partem da combinação de funções simples e triviais para obter uma seleção complexa, aguçada e especializada em certa característica, isto é, a cada camada da rede que se percorre, uma pequena característica é analisada (bordas, tons de cores) e a coleção dessas análises permite conclusões complexas (tipo de objeto, por exemplo).

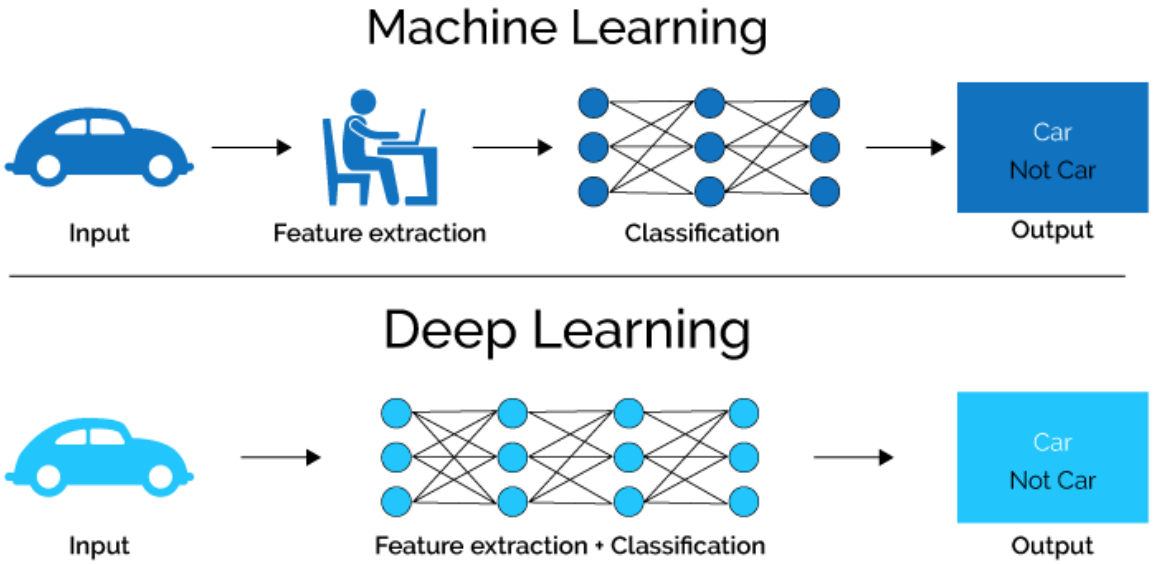


Figura 16: Comparativo entre *deep learning* e *machine learning* [16]

A aprendizagem profunda é parte de uma família mais abrangente de métodos de aprendizado de máquina baseados na aprendizagem de representações de dados. Uma observação (por exemplo, uma imagem), pode ser representada de várias maneiras, tais como um vetor de valores de intensidade por pixel, ou de uma forma mais abstrata como um conjunto de arestas, regiões com um formato particular, etc. Algumas representações são melhores do que outras para simplificar a tarefa de aprendizagem (por exemplo, reconhecimento facial ou reconhecimento de expressões faciais). Uma das promessas da aprendizagem profunda é a substituição de características feitas manualmente por algoritmos eficientes para a aprendizagem de características supervisionada ou semi-supervisionada e extração hierárquica de características.

A pesquisa nesta área tenta fazer representações melhores e criar modelos para aprender essas representações a partir de dados não rotulados em grande escala. Algumas das representações são inspiradas pelos avanços da neurociência e são vagamente baseadas na interpretação do processamento de informações e padrões de comunicação em um sistema nervoso, tais como codificação neural que tenta definir uma relação entre vários estímulos e as respostas neuronais associados no cérebro.

Várias arquiteturas de aprendizagem profunda, tais como redes neurais profundas, redes neurais profundas convolucionais, redes de crenças profundas e redes neurais recorrentes têm sido aplicadas em áreas como visão computacional, reconhecimento automático de fala, processamento de linguagem natural, reconhecimento de áudio e bioinformática, onde elas têm se mostrado capazes de produzir resultados do estado-da-arte em várias tarefas.

4.1.3 Teste e Validação

Para validar o modelo normalmente divide-se os dados em dois *datasets* distintos, um para treinar o modelo, por exemplo 70% do total, e os outros 30% para testá-lo. É bem importante realizar essa abordagem para conseguir de fato uma validação do modelo, já que a correta validação da generalização do modelo deve ser feita por um conjunto de dados desconhecido. Outra técnica comumente utilizada para validação é a de validação cruzada com *K-Fold* que permite continuar utilizando uma grande quantidade de dados e ao mesmo tempo gera boa quantidade de dados para teste do modelo. O seu funcionamento é baseado na criação de *k subsets*, e para cada treinamento do modelo um dos *subsets* é utilizado como conjunto de teste e os outros *k - 1 subsets* são utilizados como conjunto de treino.

No final a estimativa de erro do modelo é gerada através da média dos erros de todos esses testes realizados. Esse tipo de validação é muito útil, pois a permutação dos dados de teste com os dados de treinamento tende a aumentar a eficácia do modelo.

4.1.3.1 Matriz de Confusão

Como pode ser visto na figura 20, em uma matriz de confusão as colunas representam as classes verdadeiras e as linhas as previstas pelo modelo. Assim o elemento aij corresponde ao total de predições que o modelo fez para a classe i mas que eram na verdade da classe j e o elemento akk representa o total de predições corretas para a classe k. Como ilustrado abaixo:

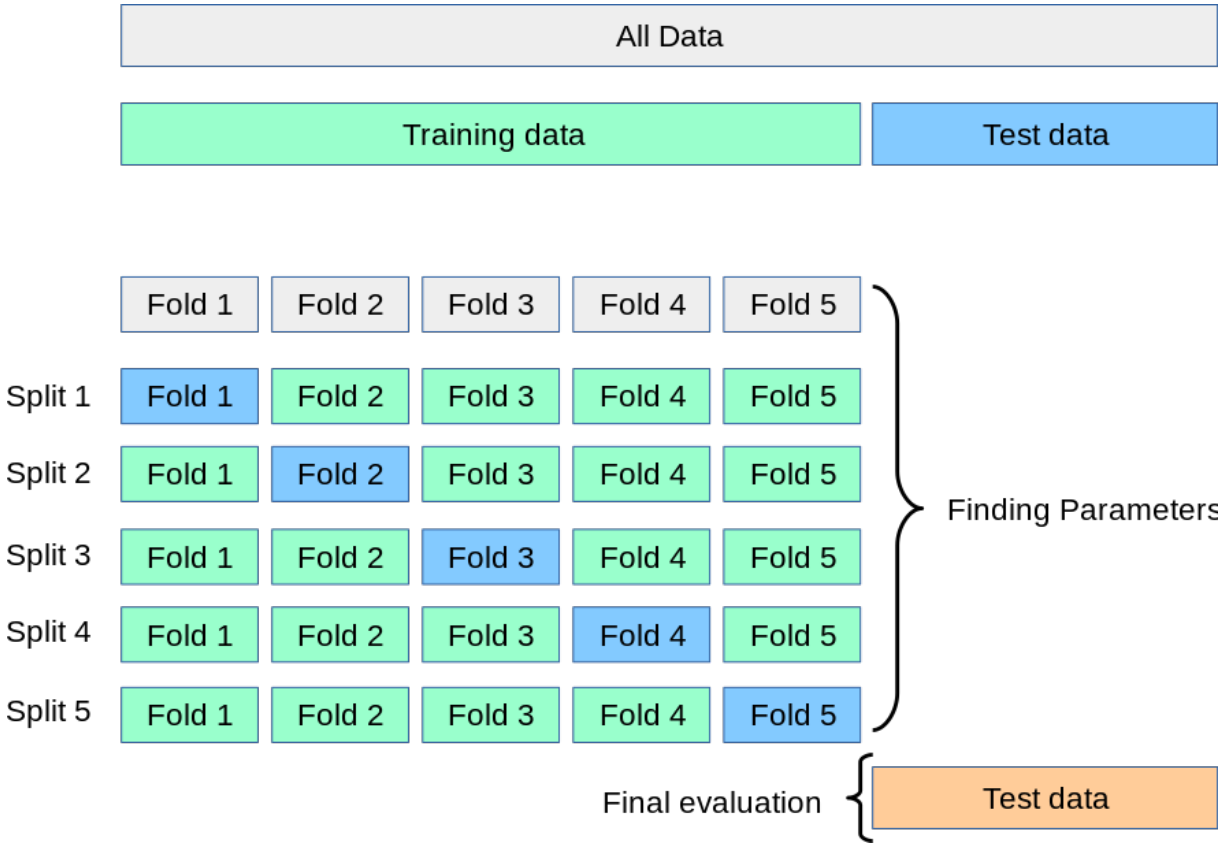


Figura 17: Esquema Validação Cruzada com 5 *Folds* [17]

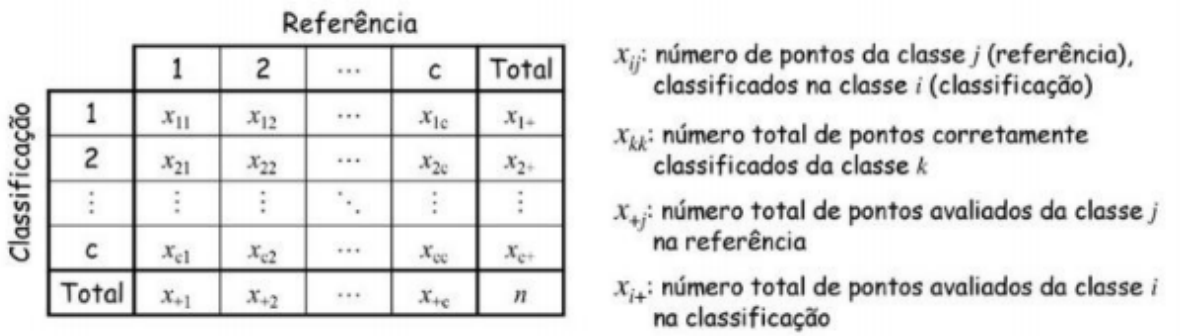


Figura 18: Matriz de Confusão

4.1.3.2 Erro

- Erro Quadrático Médio :

$$RMSE(X,h) = \sqrt{\frac{1}{/}m\sum_{n=1}^{m}(h(x^n) - y^n)^2}$$

- Erro Absoluto Médio:

$$MAE(X,h) = \frac{1}{/}m \sum_{n=1}^{m} |(h(x^n) - y^n)^2|$$

Sendo em ambas as equações:

- X : matriz contendo todas as *features* e *labels* correspondentes
- h : hipótese do sistema, ou seja, o valor da predição do modelo
- m : número de instâncias no *dataset*
- x^n: vetor com todos os valores das *features*
- y: *labels* do sisetma

4.1.3.3 Acurácia

Indica uma performance geral do modelo. Dentre todas as classificações, quantas o modelo classificou corretamente. Utilizando a matriz da figura 20 como referência temos:

$$Acurácia = \frac{\sum_{k=1}^{c} x_{kk}}{n}$$

4.1.3.4 Precisão

É uma métrica específica para cada classe, ela mede a quantidade de acertos classificados como pertinentes à categoria, ou seja, a quantidade de classificações acertadas da classe pelo modelo sobre o total de predições da classe. Utilizando a matriz da figura 20 como referência, temos que a precisão da classe k é dada por:

$$Precisão = \frac{x_{kk}}{x_{+k}}$$

4.1.3.5 *Recall*

Também é específico de cada classe, essa métrica mede a quantidade de acertos pertencentes à categoria, ou seja, a quantidade de classificações acertadas da classe pelo modelo sobre o total real da classe. Utilizando a matriz da figura 20 como referência, temos que o *recall* da classe k é dada por:

$$Recall = \frac{x_{kk}}{x_{k+}}$$

4.1.3.6 *F-score*

É a média harmônica das medidas de precisão de *recall*. Com P: Precisão e R: *Recall*, temos:

$$F - score = \frac{2PR}{P + R}$$

4.2 Linguagem e Bibliotecas

Para o desenvolvimento de aplicações de ciência de dados a linguagem de programação *Python* tem se destacado por sua facilidade de escrita, muito em função de ter tipagem forte e dinâmica. A tipagem forte para uma aplicação em dados é uma característica importante, pois não permite que sejam realizadas operações com variáveis totalmente diferentes, o que muito provavelmente não faria sentido em uma análise estatística. Ao mesmo tempo, a tipagem dinâmica permite a concentração do desenvolvedor no projeto propriamente dito ao invés da sintaxe e características da linguagem. Além disso, algumas ferramentas surgiram como o *Jupyter Notebook* que permite a rápida visualização e iteração com os dados. Dentre os principais projetos e bibliotecas do *Python*, algumas possuem maior destaque:

- ***Scikit-Learn***: Biblioteca que possui uma série de modelos e métodos de avaliação de performance já implementados e prontos para o uso. É uma das bibliotecas de código aberto mais importantes na comunidade de desenvolvimento, estando em constante processo de melhoria e com algoritmos com documentações completas e organizadas, permitindo o rápido avanço em projetos de inteligência artificial.

- ***Jupyter Notebook***: Ferramenta interativa que permite um ambiente de desenvolvimento via navegador, sendo importante para exploração analítica de dados e facilidade para criação de projetos documentados, com código, texto e imagens, facilitando sua disseminação com integrantes de um projeto.

- ***NumPy***: É um dos principais pacotes para projetos de ciência de dados. Ele contém ferramentas de randomização de dados, permite trabalhar com vetores multidimensionais, além de possuir várias funções matemáticas.

- ***SciPy***: É uma coleção de funções de ciências de dados, permitindo operações avançadas de álgebra linear, distribuições estatísticas, sendo um dos principais componentes do *Scikit-Learn*.

- ***Matplotlib***: É uma das bibliotecas básicas que permite a criação de gráficos. Apesar de parecer secundário, a análise de dados de forma visual é uma das mais importantes ferramentas em projetos de *Machine Learning*, pois permite a identifcação de *insights* que não seriam percebidos apenas observando os números em sua forma bruta.

- **Pandas**: É uma biblioteca para análise de dados em forma de tabelas, similar ao excel. O pacote permite modificar e trabalhar com os dados de maneira dinâmica, com *queries* similares com as de SQL, além de permitir a ingestão de dados provenientes de bases SQL, excel e CSV's.

- **GeoPandas**: É uma biblioteca *open source* para facilitar o trabalho com dados geoespaciais em *python*. O GeoPandas estende os tipos de dados usados pelos pandas para permitir operações espaciais em tipos geométricos e plotar mapas espaciais.

- ***Bokeh***: É uma biblioteca de visualização interativa para navegadores modernos. Ele fornece uma construção elegante e concisa de gráficos versáteis e oferece interatividade de alto desempenho em grandes conjuntos de dados ou *streaming*.

- ***Seaborn***: É uma biblioteca de visualização de dados *Python* baseada no *matplotlib*. Ele fornece uma interface de alto nível para desenhar gráficos estatísticos atraentes e informativos.

PARTE II

EXPERIMENTOS

5 EXPERIMENTO INICIAL - HELLO WORLD

O primeiro ciclo do processo de *Big Data* se concentrou ao redor da replicação reduzida de Suel et Al (2019) [1]. O código da replicação está disponível em `https://gitlab.com/satelitaltcc/image-gatherer`.

5.1 Aquisição dos Dados Socioeconômicos

A primeira fase da replicação consistiu em obter os dados assim como descrito no artigo. Foram baixados os dados do censo UK 2011 [1]. Por uma questão de simplicidade e de tempo, optou-se por não utilizar todas as variáveis socioeconômicas descritas, mas somente as que estavam diretamente relatadas no censo:

- *General health*
- *Occupancy rate*
- *Unemployment*

Os resultados de cada índice socieoeconômico estão em decils, no qual o 1 decil corresponde aos 10% piores da amostra e o 10 decil aos 10% melhores da amostra.

5.2 Aquisição das Imagens do *Google Street View*

O artigo menciona muito brevemente a aquisição de imagens do Google Street View. Embora citem a API de imagens do Street View[2], não fornecem códigos para uma replicação completa. Além disso, a API possui um custo que, embora baixo para cada imagem, torna-se impraticável dado a elevada quantidade de imagens necessária — só em

[1] `https://www.ons.gov.uk/census/2011census/2011censusdata`
[2] `https://developers.google.com/maps/documentation/streetview/intro`

Londres, utilizou-se um total de 525,860 imagens, o que inviabiliza o uso da fonte descrita no artigo. Diante disso, havia três possibilidades:

1. Utilizar outro serviço gratuito análogo ao Street View – InstantStreetView[3]. Esse serviço mostrou-se lento e instável para a quantidade de imagens necessária;

2. Utilizar o serviço de panorama estático do Street View. Esse serviço retorna para cada localidade uma imagem associada. Infelizmente, esse serviço não permite escolher parâmetros relacionados à câmera; isto é, ele retorna somente uma imagem por localidade. Dessa forma a replicação seria fundamentalmente diferente do artigo original, uma vez que este usa quatro imagens por localidade (0º, 90º, 180º, 270º);

3. Utilizar o CBK do Google Maps, isto é, o servidor de imagens para o Google Maps. Esse servidor retorna para uma dada localidade uma imagem panorâmica, com projeção cilíndrica. Há um parâmetro *zoom* que permite, junto com os parâmetros x, y obter um segmento da imagem, reduzindo a distorção.

A última foi a escolhida para obtenção das imagens. Há um *trade-off* entre distorção e quantidade de imagens necessárias. Quanto maior o *zoom*, menor a distorção, mas mais imagens são necessárias para montar a vista. Neste trabalho temos escolhido o valor de zoom = 3

(a) (b) (c) (d) (e)

Figura 19: Exemplos de imagens do Google Maps CBK com zoom variável de 1 a 5

5.3 Extração de *Features* das Imagens

Como mencionado no item 3.1.2.2 *Deep Learning* (Aprendizagem Profunda), a extração das *features* em um modelo de *Deep Learning* é feito pela própria máquina.

Assim, para extrair um vetor de características latentes das imagens provenientes do *Google Street View* utilizou-se a mesma rede neural que utilizaram no artigo, a VGG16, a qual está disponível na aplicação *ImageNet*.

[3] `https://www.instantstreetview.com/`

5.3.1 *VGG16* e *ImageNet*

VGG16 é uma arquitetura de rede neural de convolução (CNN) que foi usada para vencer a competição *ILSVR* (*ImageNet*) em 2014. É considerada uma das principais arquiteturas de modelo computacional de imagens até hoje. O que podemos destacar sobre o VGG16 é que em vez de ter um grande número de hiperparâmetros, os desenvolvedores da rede concentraram-se em ter camadas de convolução 3x3 com um passo de filtro e camadas *maxpool* 2x2 com dois passos de filtro para as camadas intermediárias da rede. Esse arranjo de convolução e camadas *maxpool* repete-se de forma consistente em toda a arquitetura. No final, a rede possui duas camadas totalmente conectadas (FC - *fully connected layers*) seguidos por uma camada de função de ativação, na qual utliza-se a função *softmax* para a saída. O 16 no VGG16 refere-se as 16 camadas que possuem pesos. Esta rede é uma rede muito grande e tem cerca de 138 milhões (aproximadamente) de parâmetros.

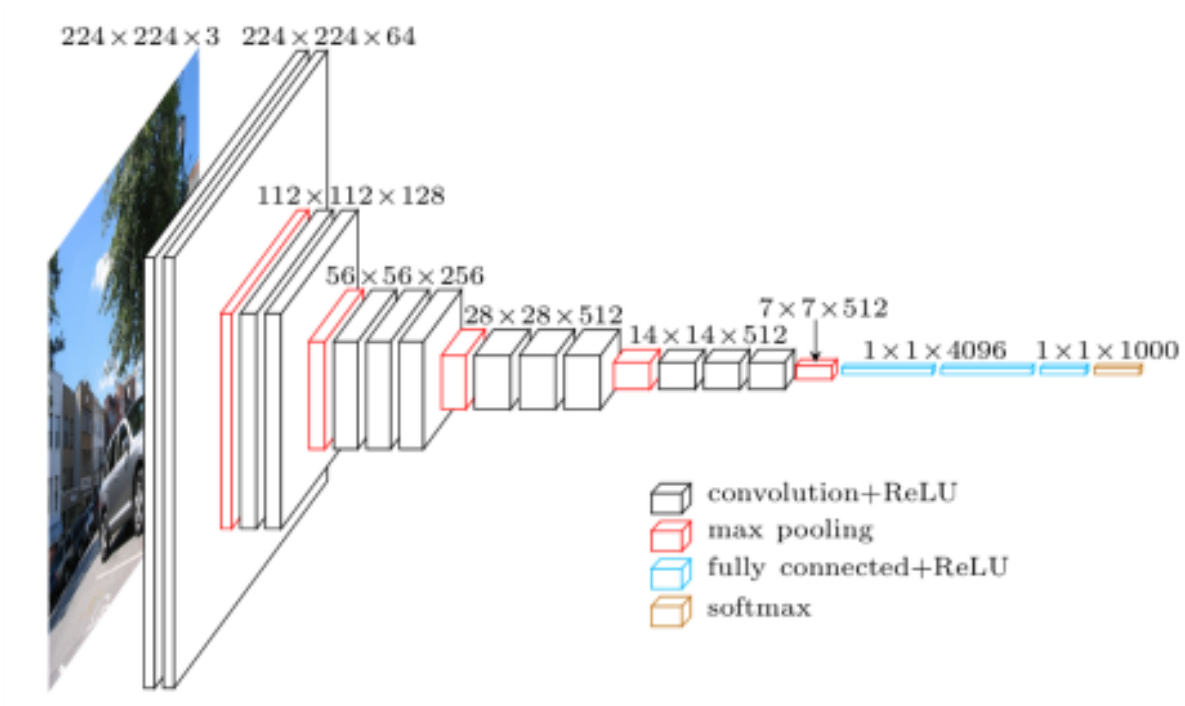


Figura 20: Arquitetura da rede convolucional VGG16. Imagem apresentada durante palestra de Mathieu Cord no evento *Deep Learning Meet up*

Assim, passou-se como entrada da rede todas as imagens adquiridas e então a VGG16 transformou-as em vetores latentes de dados. E, então, concatenou-se todos esses vetores em vários arquivos csv's, os quais serão utlizados como entrada do modelo de rede neural.

5.4 Modelo e Treinamento

O modelo recebe como entrada quatro vetores, provenientes da VGG16, que correspondem às quatro orientações de cada localidade (0º, 90º, 180º, 270º). Após passarem por três camadas, os vetores são somados e passam por mais duas camadas. A saída do modelo é unidimensional, o que significa que para cada variável socioeconômica produz-se e treina-se um modelo diferente.

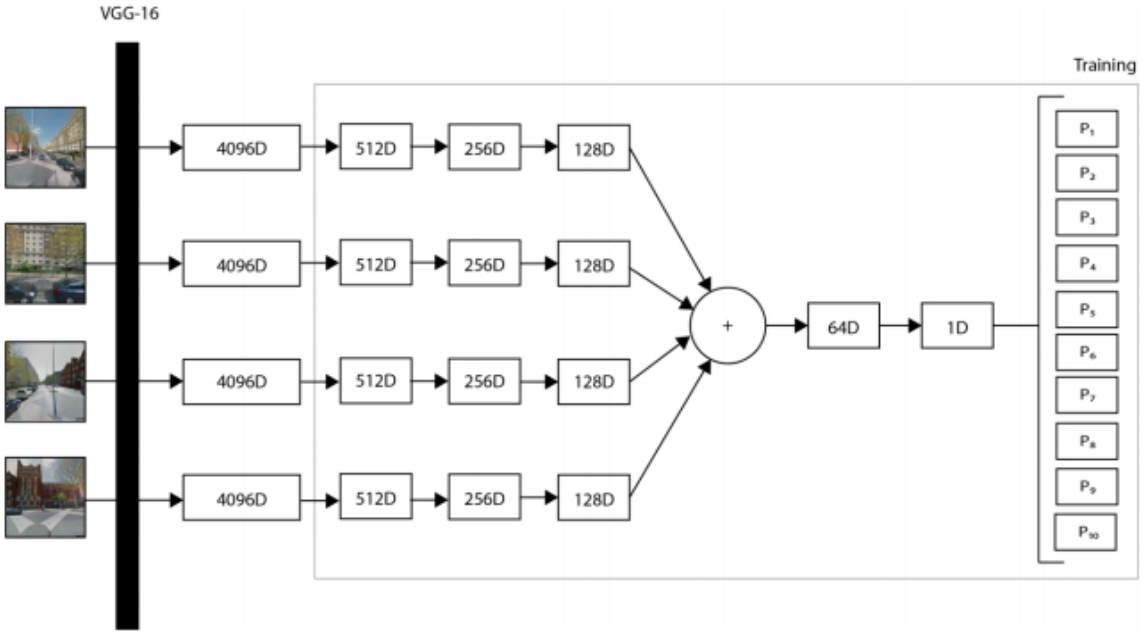


Figura 21: Suel et Al (2019) - Arquitetura do Modelo proposto por [1]

O método de validação cruzada foi o K-fold, com $K = 5$. As relações entre os folds e cada um dos elementos foram feitas randomicamente e estão presentes no repositório. O modelo foi feito usando o mesmo código do original. Por uma questão de economia de tempo e de falta de recursos computacionais, optou-se por alterar o valor do *BATCH SIZE* de 10 para 50, e reduziu-se o número de épocas para 4.

5.5 Resultados

A tabela 4 mostra as acurácias de classificação em deciles no conjunto de teste tanto no artigo original quanto na replicação. A figura 22 mostra para cada uma das variáveis analisadas o seu respectivo valor em cada uma das LSOA's de Londres, e a média de valores previstos pelo modelo de replicação.

Caso	Original			Replicação		
Margem	±0	±1	±2	±0	±1	±2
General Health	0.256	0.610	0.820	0.198	0.477	0.678
Occupancy Rating	0.307	0.666	0.858	0.299	0.634	0.817
HRP Unemployment	0.255	0.590	0.800	0.206	0.511	0.708

Tabela 1: Comparação de acurácias entre o estudo original e a replicação feita

5.6 Discussões

Analisando todas as visualizações da seção 4.4 e, principalmente a tabela de acurácias podemos concluir que os resultados ficaram um pouco abaixo dos apresentados pelo artigo original. Os mapas de calor da cidade de Londres das variáveis *general health* e *unemployment* não ficaram tão bons, não conseguiram diferenciar tão bem as especificidades de cada região de Londres, resultando em mapas bem mais homogêneos daqueles com os valores reais. Já o mapa da variável *occupancy rating* ficou bem mais parecido com o mapa observado e com o mapa do artigo, provavelmente ao fato de a variável apresentar níveis de acurácia mais altos que as outras duas. Além disso a correlação das classes que no artigo tinham conseguido valores razoavelmente altos, na replicação ficaram bem mais baixos. Pode-se atribuir esses resultados não tão satisfatórios a três fatores: imagens utilizadas, o número de épocas usadas e o *batch* utilizado no treinamento. No treinamento da replicação o total de épocas da rede neural e o *batch* utilizado foram, respectivamente, 4 e 10. Já no artigo utilizou-se 10 épocas e um *batch* de 50. Essa redução ocorreu devido à ineficiência do computador utilizado. Além disso, as imagens utilizadas, como explicado, não foram exatamente as mesmas do artigo, essas possuíam uma distorção causada pelo tipo de projeção da API utilizada que era sem custo. Assim, acredita-se que esses fatores foram cruciais para a menor performance da replicação comparada com a do artigo e, portanto, para melhorar os resultados obtidos dever-se-ia alocar uma quantidade razoável de recursos financeiros para conseguir as imagens da API do street view e uma máquina potente na AWS. Logo, devido às circunstâncias citadas, podemos concluir que o experimento foi bastante satisfatório e os resultados apresentados foram consideravelmente bons.

5.7 Principais Problemas na Replicação

- **Imagens**: como não foi possível obter as imagens diretamente da *Google Street View Static API*, o método escolhido acaba distorcendo levemente as imagens. Isso

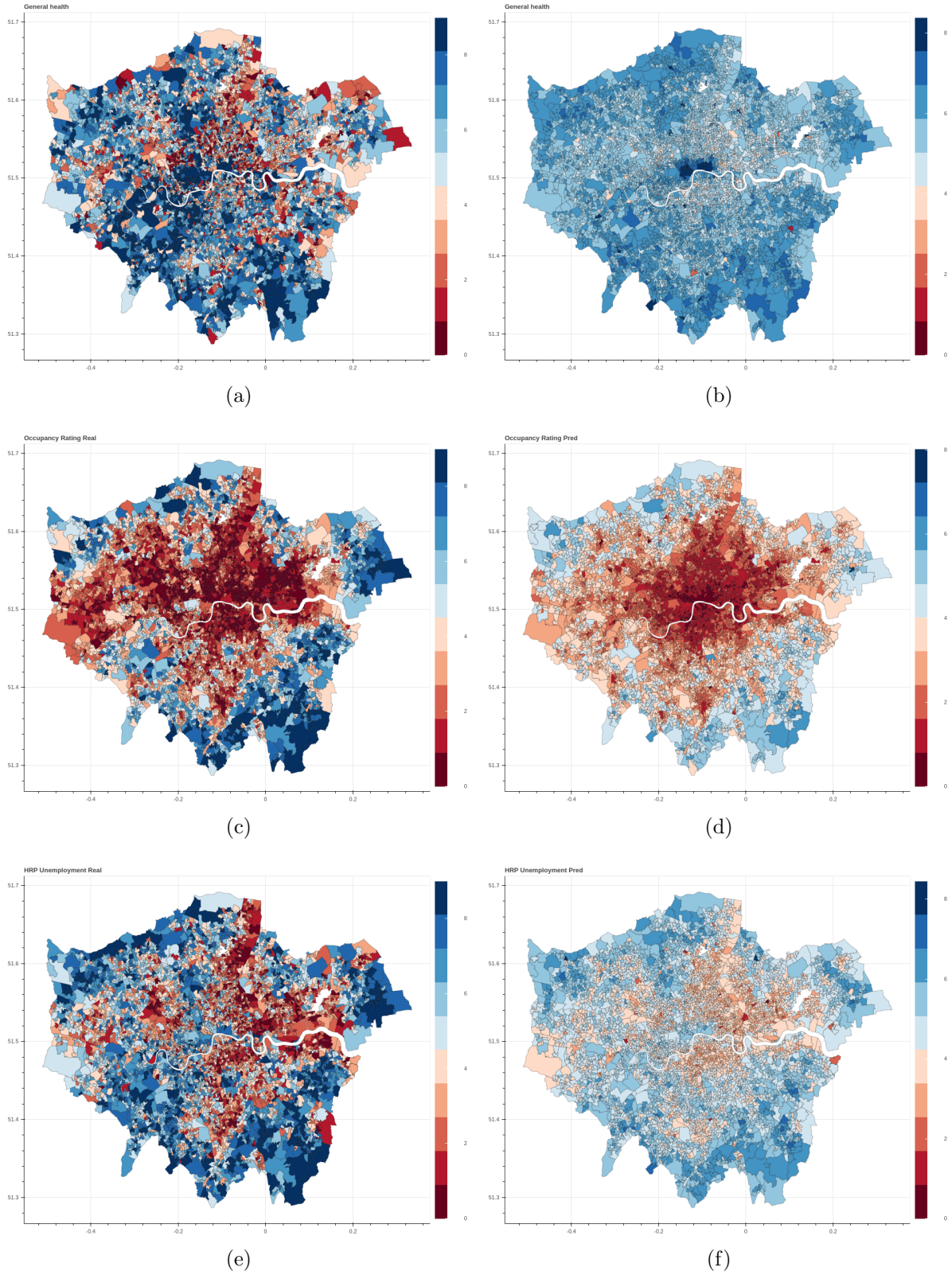


Figura 22: *Heatmap* com os deciles reais e previstos para cada uma das variáveis analisadas a) *General Health* - real, b) *General Health* previsto, c) *Occupancy rating* - real, d) *Occupancy rating* - previsto, e) *Unemployment* - real, f) *Unemployment* - previsto

pode ter trazido um certo prejuízo para o modelo de classificação;

- **Batch Size**: por uma questão de recursos computacionais e de tempo, o `BATCH SIZE` foi aumentado e o número de épocas diminuido, diminuindo a performance do modelo;

6 EXPERIMENTO VALE DO RIBEIRA

No segundo ciclo de *Data Science* aplicou-se à metodologia utilizada na replicação do artigo [1] para dados brasileiros, mais especificamente para região do vale do Ribeira nos estados de São Paulo e Paraná.

6.1 Aquisição das Imagens do *Google Street View*

As imagens que utilizou-se são imagens de ruas obtidas pelo *Google Street View*. Para aquisição destas, inicialmente programou-se um *crawler* para obter uma lista de todas as ruas dos 31 municípios do Vale do Ribeira. E com isso, conseguiu-se chamar a API do *Google Street View* que fornecia as coordenadas geoespaciais e as imagens de todas as localizações encontradas. E com base na malha de setores censitários, identificou-se à qual setor a rua pertencia e, então, conseguiu-se associar cada imagem a um setor censitário (unidade territorial estabelecida para fins de controle cadastral, formado por área contínua, situada em um único quadro urbano ou rural, com dimensão e número de domicílios que permitam o levantamento por um recenseador).

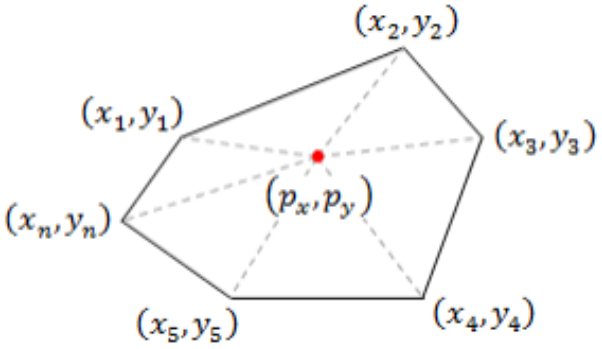


Figura 23: Mapeamento setores censitários

Porém, durante o levantamento das imagens surgiram alguns problemas. Ao fazer esse experimento, tivemos uma adaptação importante com relação a [2]: as localidades eram buscadas por endereço e num primeiro momento fizemos exatamente dessa forma.

Entretanto, tivemos um problema ao encontrar muito poucas imagens, e com uma região de abrangência bem limitada do Vale do Ribeira. Isso se deve a dois motivos principais:

- por ser uma região pouco urbanizada, há relativamente poucas ruas;
- em algumas regiões há a predominância de estradas e, ao se buscar pela localidade, a API do *Google Street View* retorna apenas um ponto, sendo que a estrada tem potencial de retornar muitos pontos em regiões diferentes de sua extensão.

A solução para essa questão foi, em vez de buscar por endereços de rua, passamos a buscar diretamente por latitude e longitude. Para cada setor censitário, obteve-se os valores de latitude e longitude mínimo e máximos desse setor, i.e., o *bounding box* do setor, e com ele montou-se uma grade de 20x20 pontos. Para cada ponto fez-se uma busca pela latitude e longitude correspondente. Dessa forma, garantiu-se que percorre-se toda a região do setor censitário, na busca por imagens. A quantidade de pontos na grade (20x20) foi determinada a partir da distância máxima d de identificação de um ponto na API do *Google Street View*. Isso significa que se existe um ponto P com imagens disponíveis, a API retorna esse ponto se a região pesquisada estiver num raio de no máximo d. Dividindo o tamanho médio dos setores censitários por d, obtivemos um valor próximo de 20. A figura abaixo exemplifica esse modo de extração de imagens, para o município de Adrianópolis. Na verdade fez-se esse procedimento para cada setor censitário, mas optou-se por mostrar a imagem do município para ilustrar mais nitidamente a grade.

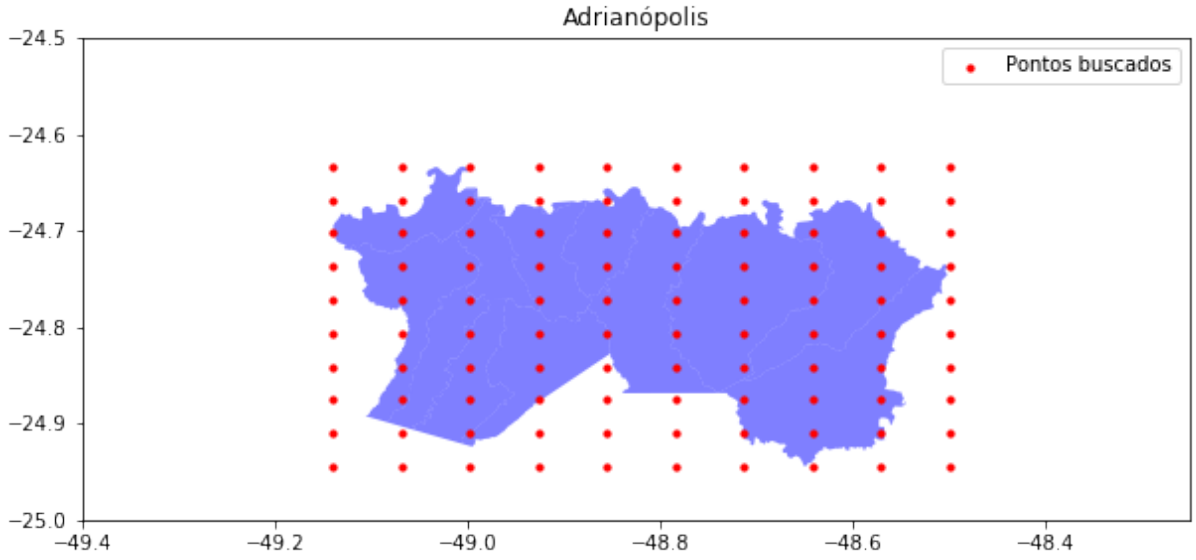


Figura 24: Exemplo de modo de extração das imagens

Com esse procedimento, conseguiu-se aumentar consideravelmente tanto o número de

imagens total quanto a região de abrangência, isto é, a região total com imagens. Mesmo assim, nem todas as regiões do Vale do Ribeira possuem imagens no *Google Street View*. Isso se deve em parte aos ambientes muito pobres, que possuem pouca infraestrutura e dificultam o acesso do carro de fotografias. Outra questão importante é a existência de algumas reservas ambientais, cujo acesso é restrito, o que dificultaria a presença de imagens. A figura 27 mostra as regiões com imagens disponíveis no *Google Street View*.

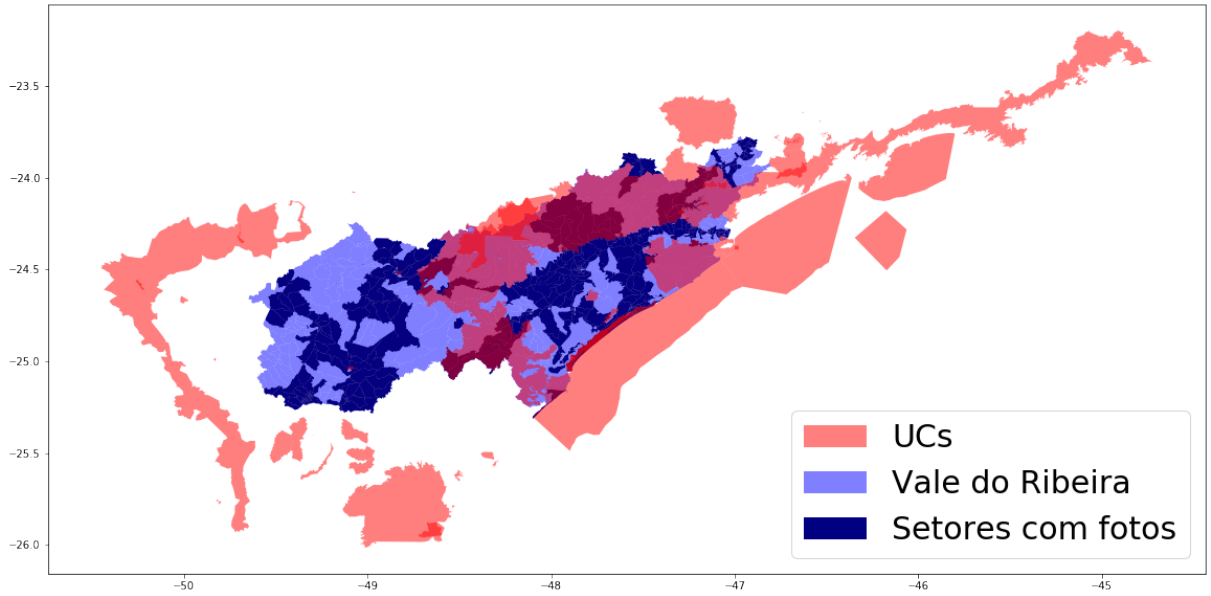
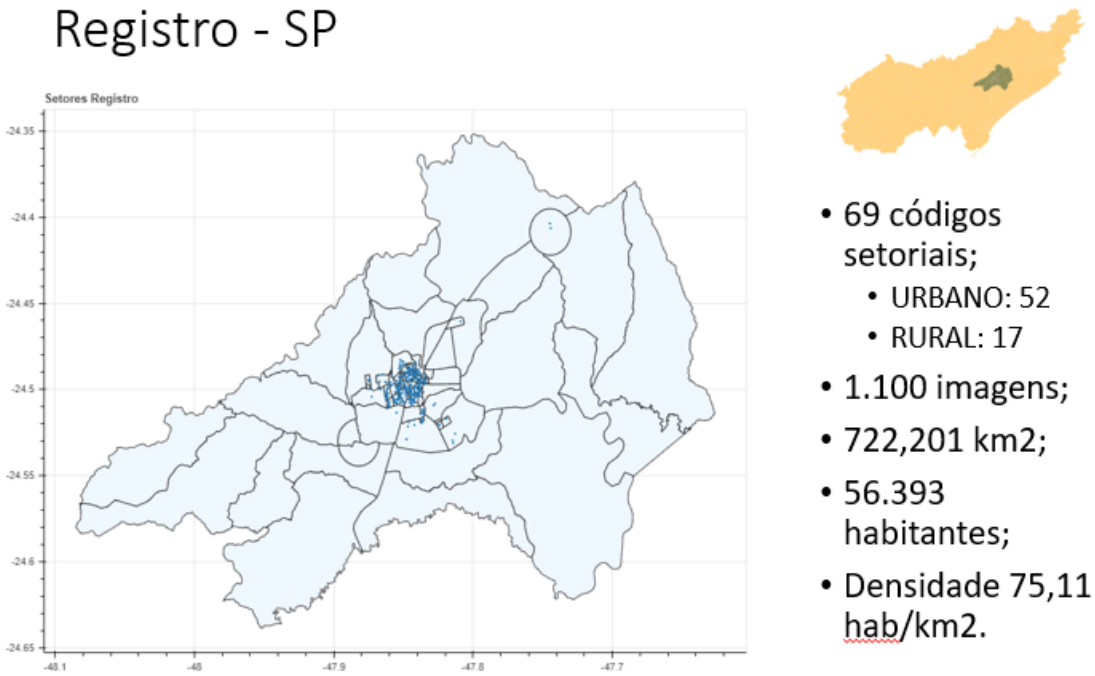


Figura 25: Disponibilidade das Imagens do GSV no Vale do Ribeira

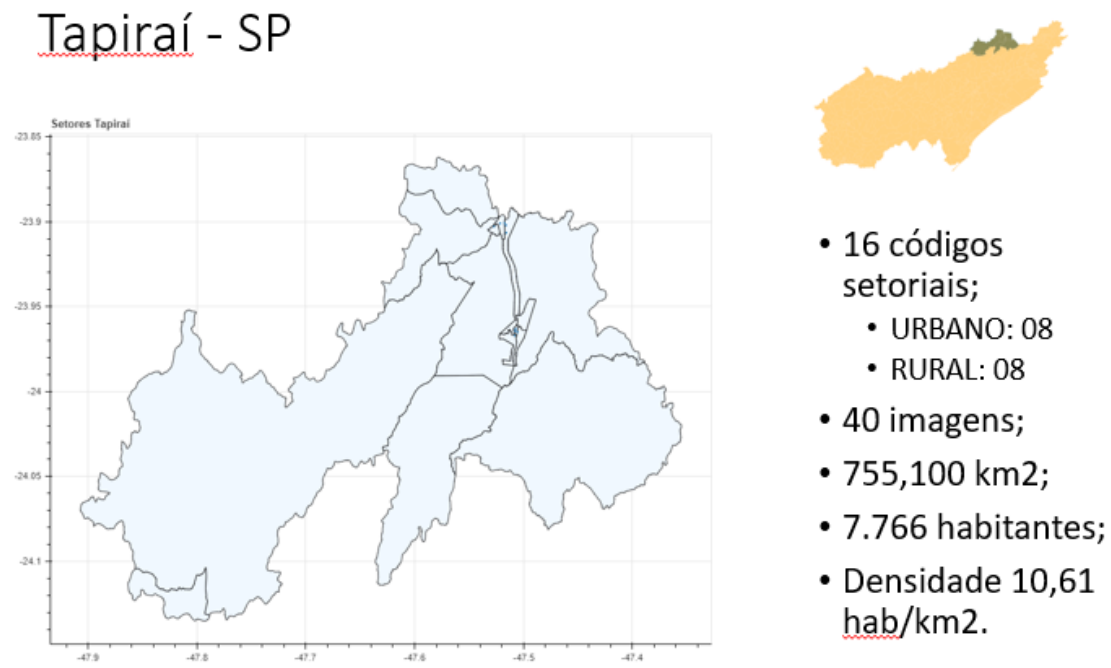
Além disso, a quantidade de imagens nos municípios é muito desbalanceada e muito concentrada em certos pontos do município, como podemos ver nas imagens da figura 28:

6.2 Aquisição dos Dados Socioeconômicos

Para treinar a rede neural precisamos de dados das respostas de cada setor censitário. No censo demográfico de 2010 disponibilizado pelo IBGE encontram-se vários indicadores sociais, porém esse trabalho focará apenas no indicador de renda. Para este indicador usou-se o levantamento "Arquivo Básico" dos conjuntos de Paraná e São Paulo Interior, filtrando posteriormente apenas os municípios pertencentes ao vale do Ribeira, que são: 'Apiaí', 'Barra do Chapéu', 'Barra do Turvo', 'Cajati', 'Cananéia', 'Eldorado', 'Iguape', 'Ilha Comprida', 'Iporanga', 'Itaóca', 'Itapirapuã Paulista', 'Itariri', 'Jacupiranga', 'Juquiá', 'Juquitiba', 'Miracatu', 'Pariquera-Açu', 'Pedro de Toledo', 'Registro', 'Ribeira', 'São Lourenço da Serra', 'Sete Barras', 'Tapiraí', 'Adrianópolis', 'Bocaiúva do Sul', 'Cerro



(a)



(b)

Figura 26: Mapas municípios de Registro e Itaraí

Azul', 'Doutor Ulysses', 'Itaperuçu', 'Rio Branco do Sul', 'Tunas do Paraná'. Para construção de um indicador de renda utilizou-se como referência o IDHM (Índice de Desenvolvimento Humano Municipal) Renda fornecido pelo Atlas Brasil, o qual calculava-se como

ilustrado abaixo:

$$\text{IDHMR} = \frac{\ln(rendaPerCapita) - \ln(valorMinimodeReferencia)}{\ln(valorMaximodeReferencia) - \ln(valorMinimodeReferencia)}$$

Calculou-se a renda per capita de cada setor censitário como ilustrado abaixo, devido às limitações das informações disponíveis na granularidade de setor censitário:

$$\text{Renda per Capita} = \frac{(rendimentoMensalMedioPorDomicílio) * (totalDeDomicilios)}{totalDeMoradoresNosDomicilios}$$

A partir da renda per capita, calculou-se decils e classificou-se os setores censitários de 1 (10% piores) a 10 (10% melhores). Na tabela abaixo tem-se a referência da faixa dos valores de renda para cada decil:

Decil	Menor Valor Renda Per Capita	Maior Valor Renda Per Capita
1	R$ 0.00	R$ 116.86
2	R$ 116.86	R$ 143.28
3	R$ 144.19	R$ 169.23
4	R$ 169.42	R$ 187.46
5	R$ 188.00	R$ 212.78
6	R$ 212.91	R$ 235.34
7	R$ 235.54	R$ 266.77
8	R$ 266.84	R$ 320.46
9	R$ 320.70	R$ 427.10
10	R$ 427.14	R$ 2500.00

Tabela 2: Faixa de valores de renda per capita para cada decil

Devido a essa abordagem não achou-se necessário a normalização utilizada pelo IDHMR com logaritmos e os valores de referência. O mapeamento de decils encontra-se ilustrado na figura 29, no qual quanto mais escura a cor, maior o decil:

6.3 Balanceamento dos decils de Renda

Um fato interessante descoberto foi a relação crescente que existe para os setores censitários entre o seu decil de renda e a quantidade de imagens presentes nesses. Isso significa que os setores censitários mais ricos possuem mais imagens, isto é, uma cobertura maior do *Google Street View*, do que os setores mais pobres. Isso se deve aos seguintes fatores:

- Há uma diferença muito grande entre ambientes urbanos e rurais, de forma que os ambientes urbanos possuem deciles maiores de renda;

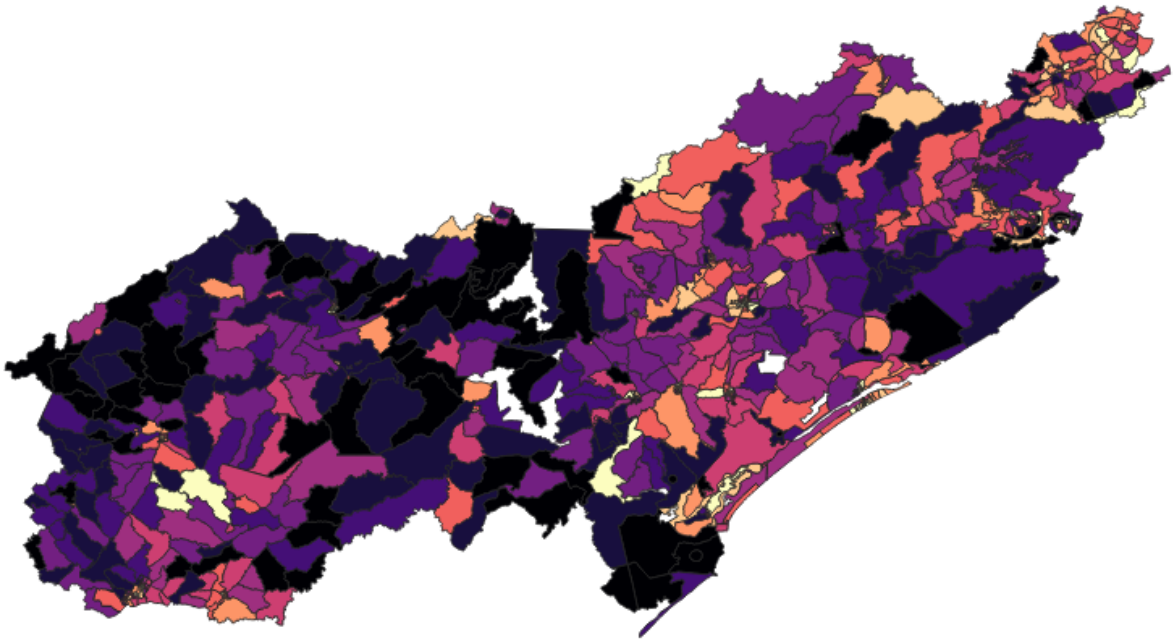


Figura 27: Vale do Ribeira - Distribuição dos Decils de Renda

- Como o *Google Street View* extrai imagens a partir das ruas, ambientes urbanos possuem mais ruas em que as imagens podem ser extraídas, enquanto ambientes rurais possuem menos ruas, predominantemente estradas.

A figura 30 mostra a distribuição da quantidade de localidades com imagens disponíveis para cada decil de renda. Existe uma relação quase de proporcionalidade entre o decil de renda e a quantidade de regiões com imagens disponíveis.

Isso atrapalharia bastante a validação do modelo, uma vez que as classes estariam desbalanceadas. Dentre as soluções possíveis para esse problema, optamos pelo *under-sampling* da menor classe, para evitar o *overfitting* e ao mesmo tempo o desbalanceamento das classes.

6.4 Extração de *Features* das Imagens

Empregou-se a mesma metodologia explicada no item 4.3, porém utlizando as imagens da região do Vale do Ribeira. Ao final, como no experimento anterior, obteve-se o vetor de características latentes das imagens.

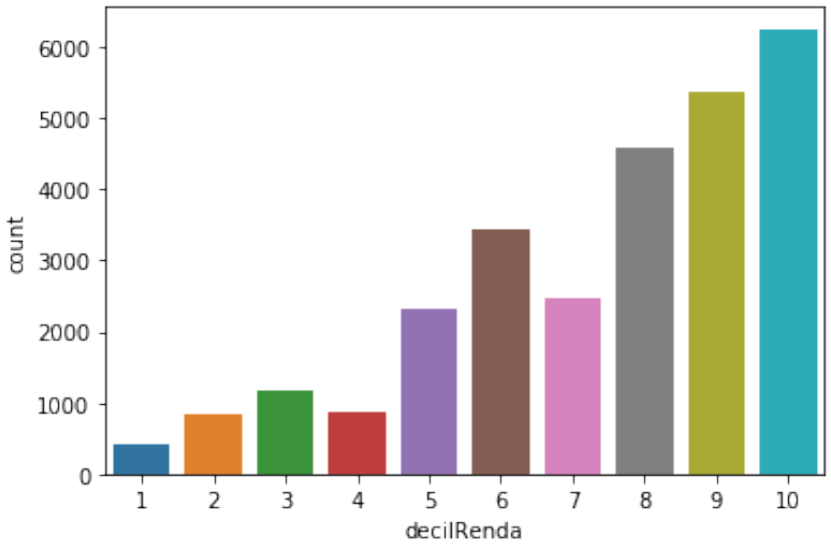


Figura 28: Contagem de regiões com imagens por decil de Renda

6.5 Modelo e Treinamento

Empregou-se a mesma metodologia explicada no item 4.4, com uma arquitetura bastante similar à apresentada na figura 21.

6.6 Resultados

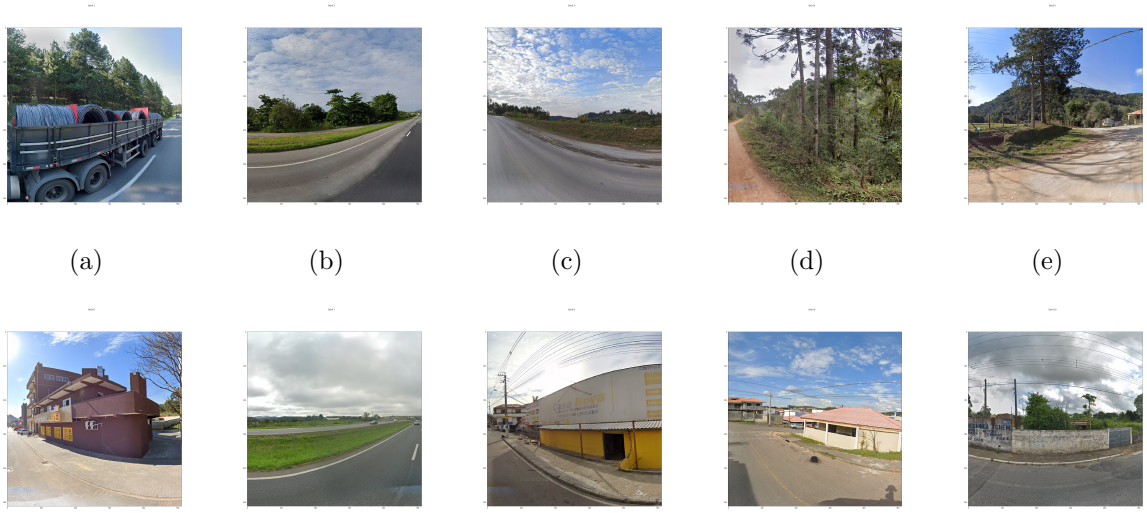
Devido ao *undersmapling* utilizado no treinamento, obteve-se como conjunto predizido apenas 156 códigos setoriais, 17% dos 914 que totalizam o Vale do Ribeira.

Uma amostra das imagens utlizadas no treinamento referentes a cada decil está ilustrado na figura 31:

A tabela 3 mostra as acurácias de classificação em deciles no conjunto de teste. A coluna ±0 representa o percentual de classes corretamente classificadas, a coluna ±1 o percentual de classes corretamente classificadas com uma margem de uma classe acima e uma classe abaixo e a coluna ±2 o percentual de classes corretamente classificadas com uma margem de duas classes acima e duas classe abaixo.

Caso	Vale do Ribeira		
Margem	±0	±1	±2
Renda per Capita	0.373	0.516	0.660

Tabela 3: Acurácias experimento Vale do Ribeira



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figura 29: Amostra das Imagens de cada decil, respectivamente do decil 1 ao 10

A estrategia utilizada no treinamento, como já mencionado, foi a de validação cruzada. E com isso, obteve-se 5 previsões de cada setor censitário. Assim, o decil predizido é a média dos 5 *folds* utlizados no treinamento.

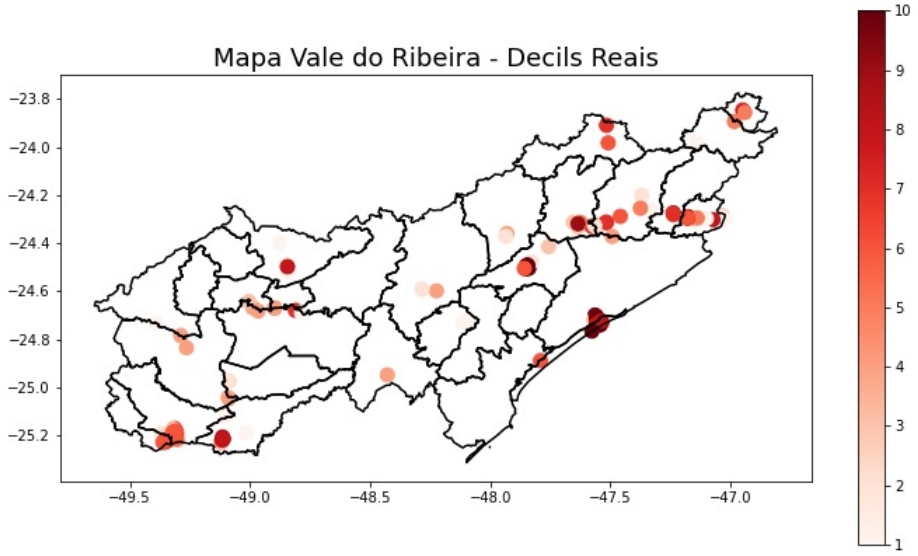
A figura 32 compara os decils reais com os decils predizidos. As marcações estão exatamente no lugar das imagens utlizadas para prever o respectivo código setorial.

Além disso, na figura 33 tem-se a média com o desvio padrão e o decil real de cada setor censitário.

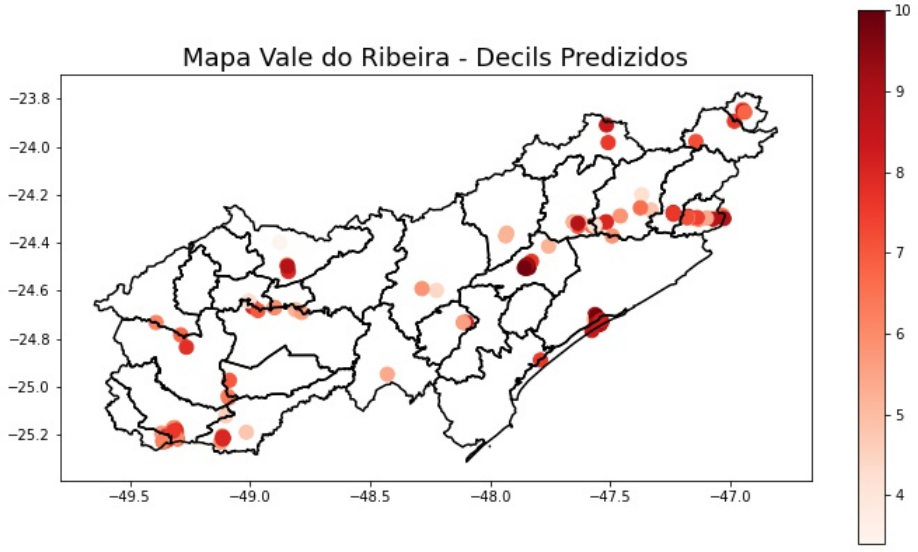
A matriz de confusão e as métricas de precisão *recall* e *F-score* para cada decil estão ilustradas nas figuras 34 e 35, respectivamente:

6.7 Comparações com o Experimento de Londres

Como detalhado no item 5.1, a aquisição de imagens do Vale do Ribeira pelo *Google Street View* foi bem mais complicada que as imagens de Londres, devido à escassez de imagens da região brasileira escolhida. Além disso, como podemos observar na figura 27, as imagens disponíveis do Vale do Ribeira estavam dispostas de maneira bastante irregular e concentrada em apenas alguns pontos. Diferentemente das imagens londrinas que eram bastante constantes e abundantes em todas as regiões da cidade. Tais fatores influenciaram consideravelmente o resultado final do experimento no Vale do Ribeira, pois enquanto que no primeiro experimento utilizou-se mais de 520 mil imagens, no experimento brasileiro menos de 1.000 imagens foram utilizadas no modelo. Apesar de tudo, os resultados do



(a)



(b)

Figura 30: Mapas Vale do Ribeira - Decils Reais e Predizidos

experimento do Vale do Ribeira foram bastante similares à replicação do experimento do artigo Suel et Al (2019) [1] e ao do próprio artigo, como podemos ver na tabela 3, a comparação de acurácias entre os resultados obtidos para renda per capita do Vale do Ribeira e para média dos 3 indicadores utilizados no experimento de Londres.

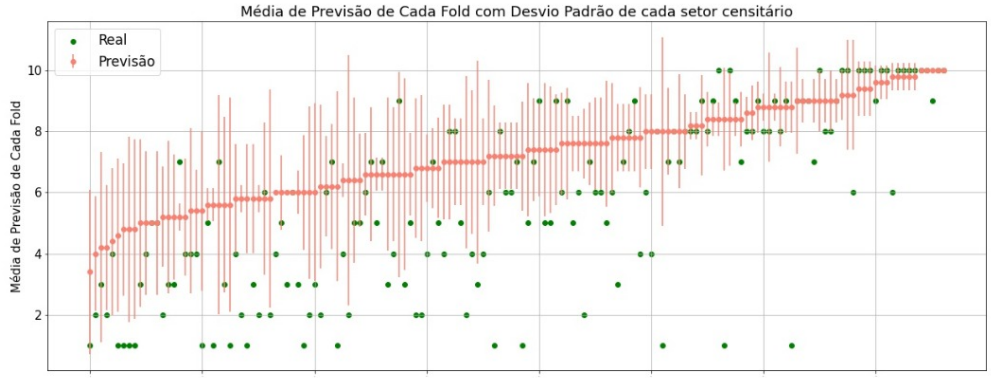


Figura 31: Média com desvio Padrão e decil real de cada setor censitário

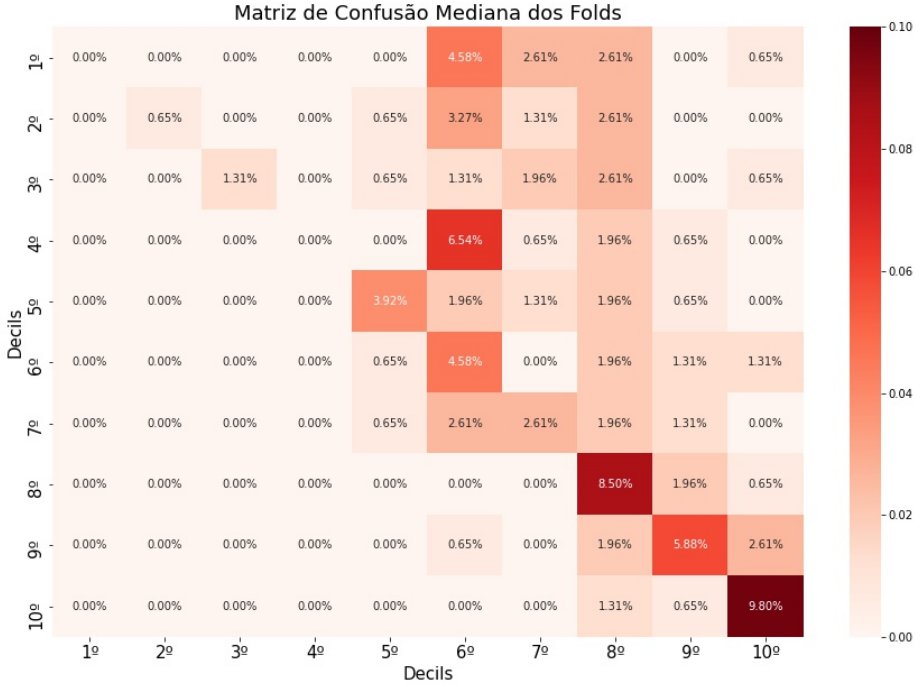


Figura 32: Matriz de Confusão

Caso	Vale do Ribeira			Londres - Original			Londres - Replicação		
Margem	±0	±1	±2	±0	±1	±2	±0	±1	±2
Valores	0.373	0.516	0.660	0.273	0.622	0.826	0.234	0.541	0.734

Tabela 4: Comparação acurácias

Analisando a tabela 4 podemos notar que o experimento do Vale do Ribeira obteve valores de acurácia mais altos para a margem 0 do que o experimento de Londres, original

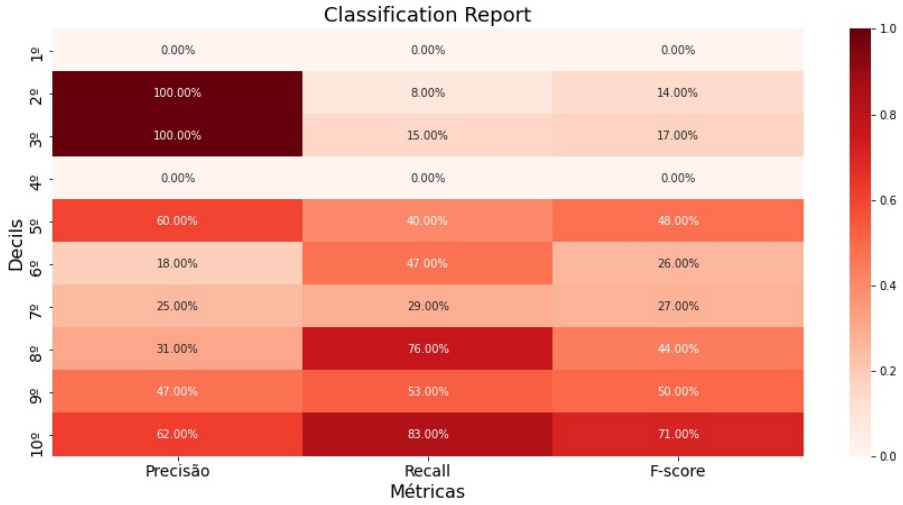


Figura 33: *Métricas de Classificação*

e replicação. Porém para as margens 1 e 2, o experimento de Londres possui valores mais elevados. Destacando-se o de margem 2 que foi bastante superior, 66% do Vale do Ribeira quanto a 82.6% de Londres - Original e 73.4% de Londres - Replicação.

6.8 Discussões

Analisando todas as visualizações do item 5.6 pode-se concluir que o experimento apresentou resultados bastante surpreendentes dado todos os percalços encontrados em seu desenvolvimento. Como podemos observar nas imagens da figura 31, os decils não possuem tanta diferença visualmente entre si, porém a rede neural conseguiu diferenciar e acertar grande parte dos setores censitários.

Observando os mapas da figura 32, pode-se dizer que em uma primeira análise, mais superficial, as predições são bastante similares aos decils reais.

Analisando a figuras 33, 34 e 35 pode-se notar que as classes mais altas, decils 8, 9 e 10, apresentaram um resultado bastante superior às outras. As classes médias, correspondentes aos decils 5, 6, 7, apresentaram resultados um pouco piores, porém na média geral. Agora as classes mais baixas, correspondentes aos decils 1, 2, 3 e 4 apresentaram resultados muito ruins. A classe 1, destaca-se negativamente, como podemos ver na figura 33, mesmo levando em consideração a média dos folds com o desvio padrão, não há nenhuma predição para essa classe. Em contraposição, a partir da classe 6 até a classe 10, levando em consideração o desvio padrão, o modelo acerta todos os casos.

No item 5.7 pode-se notar que os resultados ficaram similares ao do experimento de Londres, mesmo com apenas 0.2% das imagens disponíveis para validação e treinamento.

6.9 Conclusões

6.9.1 Considerações para um novo ciclo de Experimentos

Apesar dos resultados bastante satisfatórios apresentados por essa primeira versão do modelo de Rede Neural, pode-se levantar algumas considerações para um possível segundo ciclo do experimento, como uma nova distribuição para os dados censitários; *oversampling* ao invés de *downsampling* dos dados; expansão da região analisada.

A distribuição dos dados censitários utilizada para o treinamento foi, como descrito em 5.1, a de decils. Porém esta mostrou-se muito desequilibrada quando cruzada com as imagens disponíveis da região, como detalhado em 5.3. Assim, uma distribuição não linear, levando em consideração a quantidade de imagens disponíveis e a renda per Capita de cada região pode mostrar-se mais eficiente. Além disso, a região do Vale do Ribeira, como várias no Brasil, é extremamente desigual. E, com isso, uma distribuição linear acaba não levando em consideração a discrepância entre certas classes e a similaridade de outras.

Ao treinar a rede neural, devido ao desbalanceamento dos dados, optou-se por utlizar a estratégia de *downsampling*. Porém, como os dados disponíveis eram bastante reduzidos, pode-se tentar utilizar uma estratégia de *oversampling*, utlizando as técnicas descritas no item 3.1.1.5 para tentar evitar o *overfitting*.

Além disso, uma possível estrategia de melhoria para o modelo seria ampliar a região analisada, devido à escassez de dados no Vale do Ribeira.

6.9.2 Principais problemas encontrados na execução do experimento

O principal problema do experimento no Vale do Ribeira foi a obtenção e a qualidade das imagens. Apesar de ter-se aplicado várias técnicas diferentes para aquisição das imagens, devido à escassez destas a parte do treinamento foi bastante afetada.

REFERÊNCIAS

[1] Esra Suel, John W. Polak, James E. Bennett, and Majid Ezzati. Measuring social, environmental and health inequalities using deep learning and street imagery. *Scientific Reports*, 9(1):1–10, 2019.

[2] Neal Jean, Marshall Burke, Michael Xie, W. Matthew Davis, David B. Lobell, and Stefano Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353:790–794, 2016.

[3] Ironhack. O que é machine learning?, 2019.

[4] Anukrati Mehta. An Ultimate Guide to Understanding Supervised Learning, 2019.

[5] CHI Software. Supervised vs. Unsupervised Machine Learning, 2019.

[6] Ironhack. Introduction to Reinforcement Learning : In 2(or a bit more) Minutes, 2019.

[7] Sai Nikhilesh Kasturi. Underfitting and Overfitting in machine learning and how to deal with it !!!, 2019.

[8] Indresh Bhattacharyya. SMOTE and ADASYN (Handling Imbalanced Data Set), 2018.

[9] Hong Man Sachi Desai Haibo He, Sheng Chen and Shafik Quoraishee. Imbalanced learning for pattern recognition: an empirical study. 2010.

[10] K. Agustianto and P. Destarianto. Imbalance data handling using neighborhood cleaning rule (ncl) sampling method for precision student modeling. pages 86–89, 2019.

[11] André Pacheco. Introdução a Redes Neurais Artificiais, 2015.

[12] Vinicius. REDES NEURAIS ARTIFICIAIS, 2017.

[13] JEREMY JORDAN. Setting the learning rate of your neural network., 2018.

[14] Shruti Jadon. Introduction to Different Activation Functions for Deep Learning, 2018.

[15] Thomas Wood. What is Backpropagation? , 2015.

[16] ED SPERLING. Deep Learning Spreads, 2018.

[17] scikit-learn developers. Cross-validation: evaluating estimator performance, 2020.

[18] Bengio Y. Hinton G. LeCun, Y. Deep Learning. *Science*, 2015.

[19] Rohit Thakur. Step by step VGG16 implementation in Keras for beginners, 2019.

[20] Sanatan Mishra. Unsupervised Learning and Data Clustering, 2017.

[21] Aidan Wilson. A Brief Introduction to Supervised Learning, 2019.

[22] Valentina Alto. Neural Networks: parameters, hyperparameters and optimization strategies, 2019.

[23] D. Oliveira C. Aridas G. Lemaitre, F. Nogueira. imblearn.under$_s ampling.AllKNN$, 2016.

[24] Conner Brew. Level Up Your Visualizations: Make Interactive Maps with Python and Bokeh, 2011.

[25] Jorge Lopez. Combining Satellite Imagery and machine learning to predict poverty, 2019.

[26] Sarang Narkhede. Understanding Confusion Matrix, 2018.

[27] Dennis T. Confusion Matrix Visualization, 2019.

[28] Kurtis Pykes. Oversampling and Undersampling A technique for Imbalanced Classification, 2010.

[29] Michael Nielsen. CHAPTER 2 How the backpropagation algorithm works, 2019.

[30] Simeon Kostadinov. Understanding Backpropagation Algorithm, 2019.

[31] Arthur Lamblet Vaz. Como lidar com dados desbalanceados em problemas de classificação, 2019.

[32] Viana. O que é Overfitting e Underfitting em Machine Learning, 2019.

[33] Ashwani Dhankhar. Mapping with matplotlib, pandas, geopandas and basemap in python.

[34] GeoPandas developers. Plotting with Geoplot and GeoPandas, 2019.

[35] Gulshan Baraik. How to Plot Mean and Standard Deviation in Pandas?, 2020.

[36] Viana. bokeh.palettes, 2019.

[37] Jason Brownlee. What is Deep Learning?, 2019.

[38] Konrad Budek Błażej Osiński. What is reinforcement learning? The complete guide, 2018.

[39] Paulo Vasconcellos. Explicando Deep Reinforcement Learning com Super Mario ao invés de matemática, 2018.

[40] Michael Waskom. seaborn: statistical data visualization¶.

[41] Bokeh Contributors. Bokeh, 2019.

[42] Data Science Academy. Capítulo 14 – Algoritmo Backpropagation Parte 1 – Grafos Computacionais e Chain Rule, 2019.

[43] Jason Brownlee. Understand the Impact of Learning Rate on Neural Network Performance, 2020.

[44] Jason Brownlee. Stacking Ensemble for Deep Learning Neural Networks in Python, 2020.

[45] Data Science Academy. Capítulo 27 – A Taxa de Aprendizado de Uma Rede Neural, 2019.

[46] Jason Brownlee. Gradient Descent For Machine Learning, 2019.

[47] Katanforoosh Kunin. Initializing neural networks, 2019.

[48] Jason Brownlee. Supervised and Unsupervised Machine Learning Algorithms, 2020.

APÊNDICE A – METODOLOGIA DE UM PROJETO DE *BIG DATA*

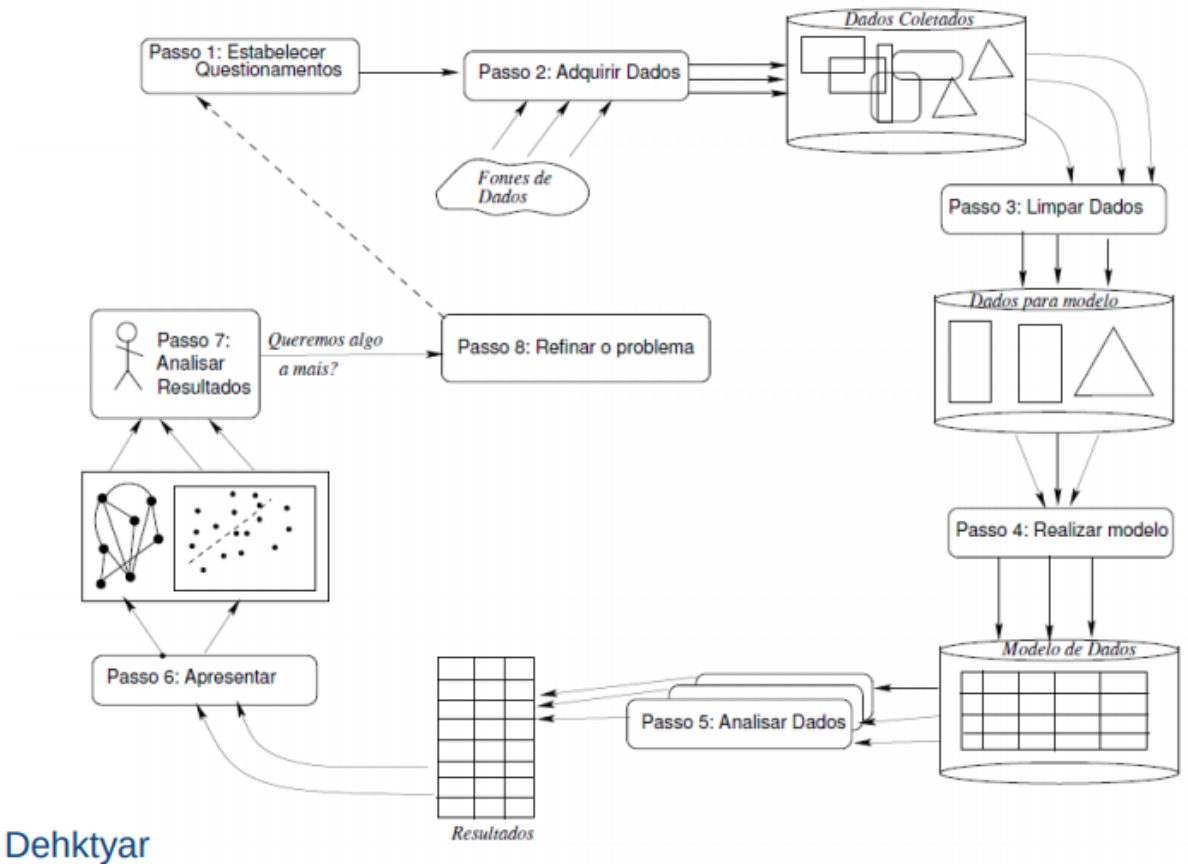


Figura 34: Metodologia de um projeto de *Big Data*