# Nanodegree Engenheiro de Machine Learning

## Proposta de projeto final

Bruno Aurélio Rôzza de Moura Campos
10 de dezembro de 2018

## Proposta

### Histórico do assunto

O aprendizado supervisionado é a área de pesquisa mais explorada dentro de machine learning, reflexo disso são as aplicações comerciais como por exemplo, árvores de decisão para sistemas de recomendação. Os usuários que consomem esses serviços comerciais já são afetados por técnicas de machine learning. Contudo, nem todos os serviços exploram seus dados gerados para obter insights e assim melhorar a qualidade para os usuários finais.
Um dos fatores que mais afeta os clientes ao consumir um serviços é o atraso. Para reduzir o período de espera, as organizações podem usar sistemas de recomendação, indicando soluções mais eficientes e rápidas. Isso não apenas ajuda os clientes, mas também permite que as empresas se concentrem em outras coisas que exigem intervenção humana regular. Como exemplo do uso desta técnica, é possível citar a empresa Linx Impulse que fornece serviço de sistemas de recomendação aos grandes e-commerces da américa latina, através de emails, vitrines e busca personalizadas para auxiliar os usuários na tomada de decisão.
Devido a estes fatores, a minha motivação pessoal é utilizar técnicas de aprendizado supervisionado para otimizar tarefas recorrentes de atendimento ao cliente e assim aproveitar melhor o tempo.
A fonte de dados será: https://www.kaggle.com/c/allstate-claims-severity/data

### Descrição do problema

A Allstate Corporation é a segunda maior seguradora de linhas pessoais nos Estados Unidos e a maior que é de capital aberto. Devido ao seu tamanho grande, eles têm que lidar com um grande número de reclamações que leva tempo quando feito por um ser humano.
Atualmente, a Allstate está desenvolvendo métodos automatizados para prever o custo e, portanto, a gravidade das reclamações. O problema é criar um algoritmo que prevê com precisão a gravidade das reivindicações. Como entrada, será recebido diferentes variáveis que os agentes examinam para decidir o status das reivindicações. Eles podem ser contínuos ou discretos. Como a variável de destino é uma quantidade contínua (o valor a ser pago ao cliente), é essencialmente uma função de regressão. Desta forma, analisando o objetivo da seguradora neste desafio, é possível notar o algoritmo fará recomendações para os usuários.
Conforme mencionado em [4], há técnicas que filtram as recomendações para garantir uma melhor acurácia. Essas técnicas explicam a acurácia e as correlações das *features*, o que será valioso para este desafio.

## Conjuntos de dados e entradas

O conjunto de dados contém 2 arquivos .csv com informações necessárias para fazer uma previsão. Eles são:
1. Variáveis em train.csv e test.csv: - **id**: o id de um par de perguntas do conjunto de treinamento - **cat1** até **cat116**: variáveis de categoria (o intervalo de valores não é fornecido, nem os nomes das colunas). - **cont1** até **cont14**: variáveis contínuas (o intervalo de valores não é fornecido, nem os nomes das colunas). - **loss**: o valor que a empresa tem que pagar por uma determinada reivindicação. Esta é a variável de destino. - Em test.csv, a perda não está presente, já que vamos prever isso.

1. Em train.csv:
    - Número de linhas = 188318
    - Número de colunas = 132
    - Altamente relevante, pois são os dados sobre os quais vamos treinar.
2. Em test.csv:
    - Número de linhas = 125546
    - Número de colunas = 131
    - Altamente relevante, pois são os dados que testamos.

## Descrição da solução

Será necessário entender a relação entre as *features* de 130 (116 + 14) com a variável *loss*. Para isso, a solução será devenvolvida utilizado a técnica de *Hybrid filtering*, [4] para melhorar os resultados do sistema. Mais especificamente, o sistema deverá utilizar a método de _ Feature-combination_ para determinar a similaridade das *features*.
Há *features* que, devido à maldição da dimensionalidade, podem resultar em *overfitting*. Para facilitar os trabalhos podemos fazer um redução de *features* usando o PCA. Também será fundamental encontrar as correlações entre as *features* para selecionar os melhores resultados.
Nestas etapa de exploração de dados será convertido valores categóricos de alfabetos em números que podem ser mais fáceis de serem processados. Em seguida, será realizado testes com modelos de machine learning para verificar qual tem melhor performace usando a divisão do Kfold e, finalmente, obteremos o erro médio quadrático.
Os modelos testados serão: regressão linear, XGBoost e Random Forest (Bagging). Para ajustar os parâmetros no XGBoost, usaremos Grid Search.

## Modelo de referência (benchmark)

A fonte de dados é provinda de uma competição do Kaggle, então o modelo de referência será escolhido como a melhor pontuação da competição para o conjunto de teste, que aparece em 1109.70772 erro médio absoluto (menor é melhor).
Seŕa testado e executado os modelos de regressão linear, XGBoost e Random Forest (Bagging) no

conjunto de testes fornecido nesta competição do Kaggle. O arquivo de submissão será enviado ao site do kaggle para verificar a pontuação. Então, também podemos comparar nosso modelo com o modelo de benchmark hospedado pela Kaggle.
Um objeto para este trabalho é ser ranqueado entre os 30% melhores resultados, ou seja menos de 1442,620036 erro da tabela.

## Métricas de avaliação

A predição do modelo para este problema pode ser avaliada de várias maneiras. Como a avaliação oficial deste projeto é feita por Kaggle usando erro absoluto médio, o mesmo será usado para avaliação de modelos.

## Design do projeto

Para desenvolver o projeto será necessário primeiramente explorar os dados fornecidos pela competição. Na sequencia será feito a limpeza dos dados, converteção das *features* categóricos de alfabetos em números. Em seguida, será feito um *feature engineer* para um conjunto de validação cruzada afim encontrar melhores correlações entre as *features*.
Depois que os dados forem modelados será testado os modelos. Primeiro modelo a ser criado é o de regressão linear, desta forma teremos uma algoritmo linear. O segundo modelo será o de XGBoost. Por fim, o terceiro modelo o será o de Random Forest, pois assim teremos um algotimo de bagging. Depois de comparar estes 3 modelos será analisado qual tem o menor erro absoluto médio, em seguida será submetido para a competição no Kaggle.
Já sobre o tunnig e evolução do modelo final, será usado duas métricas, a acurácia e precisão do modelo.

## Bibliografia

[1] Kaggle, ”Allstate Claims Severity” (2016). https://www.kaggle.com/c/allstate-claims-severity, acessado em 18/12/2018.

[2] Machinelearningmastery, "Bagging and Random Forest Ensemble Algorithms for Machine Learning",(2016). https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/, acesso em 10/12/2018.

[3] aws, "Algoritmo XGBoost"(2018). https://docs.aws.amazon.com/pt_br/sagemaker/latest/dg/xgboost.html, acesso em 10/12/2018.

[4] Egyptian Informatics Journal, "Recommendation systems: Principles, methods and evaluation" (2015). https://ac.els-cdn.com/S1110866515000341/1-s2.0-S1110866515000341-main.pdf?_tid=0168c993-8aaa-4d01-a5c3-2cc49b72e2c1&acdnat=1545258155_05c496d643f6247ca0125c2fa2ef55ef, acesso 19/12/2018