



[← Back to Machine Learning Engineer Nanodegree](#)

Finding Donors for CharityML

REVISÃO

HISTORY

Requires Changes

2 ESPECIFICAÇÕES NECESSITAM DE MUDANÇAS

Olá,

Ótimo projeto! Apenas 2 detalhes e você conclui.

Continue com o bom trabalho 🌟

Atenciosamente,

Exploring the Data

Student's implementation correctly calculates the following:

- Number of records
- Number of individuals with income >\$50,000
- Number of individuals with income <=\$50,000
- Percentage of individuals with income > \$50,000

Você calculou corretamente as estatísticas dos dados. Bom trabalho!

Se você observou as estatísticas, está claramente indicando que as classes (indivíduos com renda > \$ 50k = 11208 e indivíduos com renda no máximo \$ 50k = 34014) estão desequilibradas. Aqui estão alguns artigos úteis

sobre manipulação de conjuntos de dados desbalanceados:

- <https://www.quora.com/In-classification-how-do-you-handle-an-unbalanced-training-set>
- <https://blog.dominodatalab.com/imbalance-datasets/>
- <https://www.kdnuggets.com/06/06/7-techniques-handle-imbalanced-data.html>

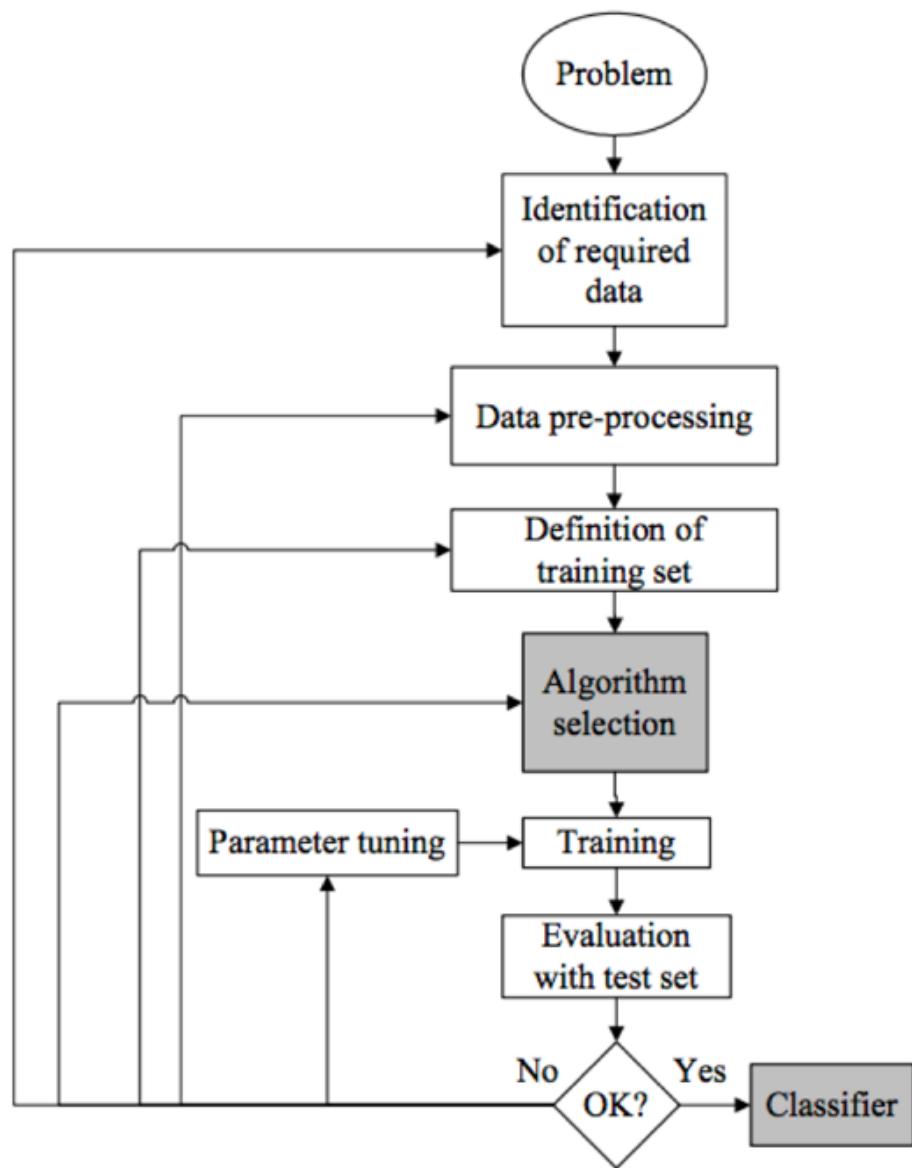
Também podemos obter algumas estatísticas iniciais com o método `pandas.describe`:

```
data.describe()
```

Outra ótima idéia seria realizar uma análise de dados exploratória. Poderia verificar a biblioteca [Seaborn](#). Por exemplo:

```
import seaborn as sns
sns.factorplot('income', 'capital-gain', hue='sex', data=data, kind='bar')
```

Você também pode encontrar no diagrama a seguir uma visão geral do processo de desenvolvimento de um modelo de Supervised Learning:



Preparing the Data

Student correctly implements one-hot encoding for the feature and income data.

Ótimo! Apenas com algumas linhas você pode aplicar as transformações.

Para sua referência, você pode conferir [este artigo que explica quando e por que usamos One Hot Encoding](#).

Algo a se notar aqui é que também podemos usar o Label Encoder como uma alternativa [Multi Class. Label Encoder](#) pode ser implementado como a seguir:

```
encoder = LabelEncoder() income = encoder.fit_transform(income_raw)
```

Sugestão: Este [link](#) fornece 7 estratégias de codificação diferentes. [Codificação binária](#), é uma ótima opção para casos em que o número de categorias é alto.

Evaluating Model Performance

Student correctly calculates the benchmark score of the naive predictor for both accuracy and F1 scores.

Ótimo trabalho calculando a precisão e o F-score para um Naive Predictor!

É sempre uma ótima idéia estabelecer um benchmark para qualquer problema. Como estes são agora considerados os nossos resultados de classificação "burros", qualquer modelo real deve ser capaz de superar esses resultados.

Observe que o F-score é maior que a precisão, o que parece ser contra-intuitivo, já que o F-score é um cálculo mais elaborado. Isso acontece porque um valor de beta = 0,5 atenua a influência de falsos negativos. Em outras palavras, esse valor de beta pesa mais as previsões positivas (> 50K) do que as negativas (<= 50K).

Dica profissional: Recomendo dar uma olhada neste [ótimo artigo para entender mais sobre como escolher a métrica certa para problemas de classificação](#).

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to be explored.

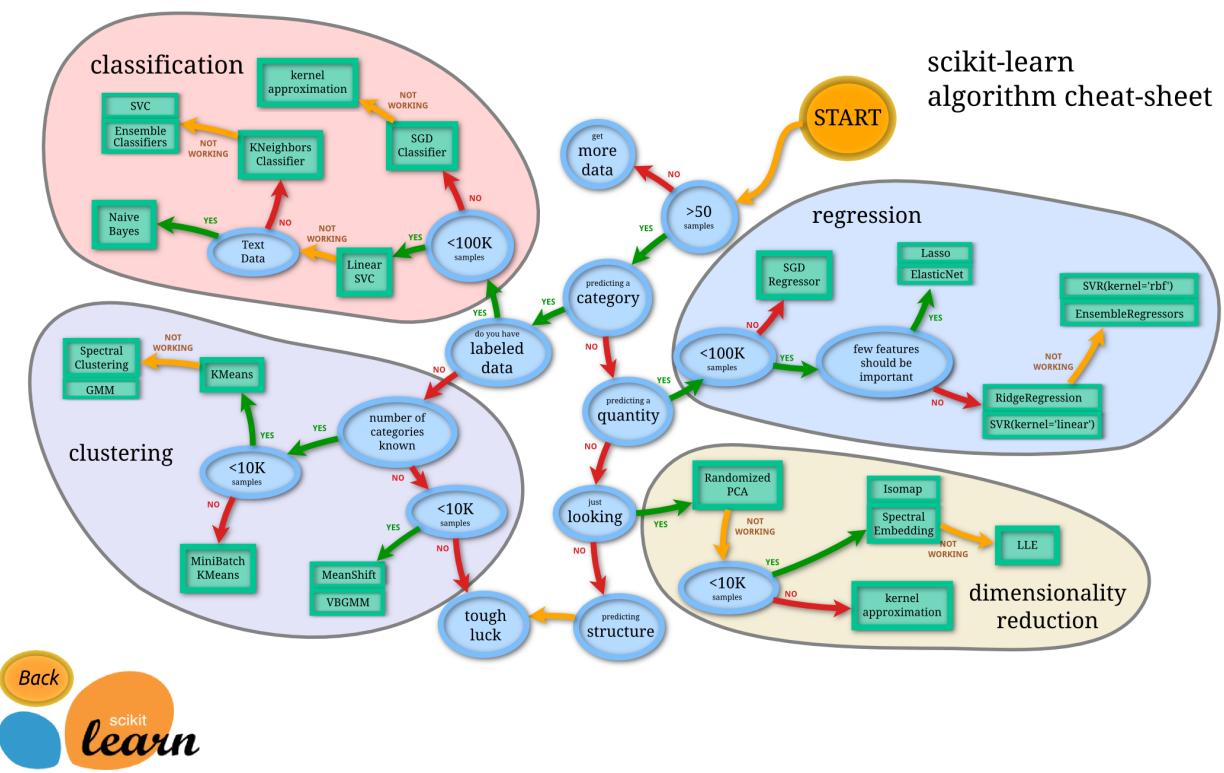
Please list all the references you use while listing out your pros and cons.

Muito bom trabalho ao mencionar algumas aplicações do mundo real, pontos fortes / fracos e raciocínio para sua escolha!

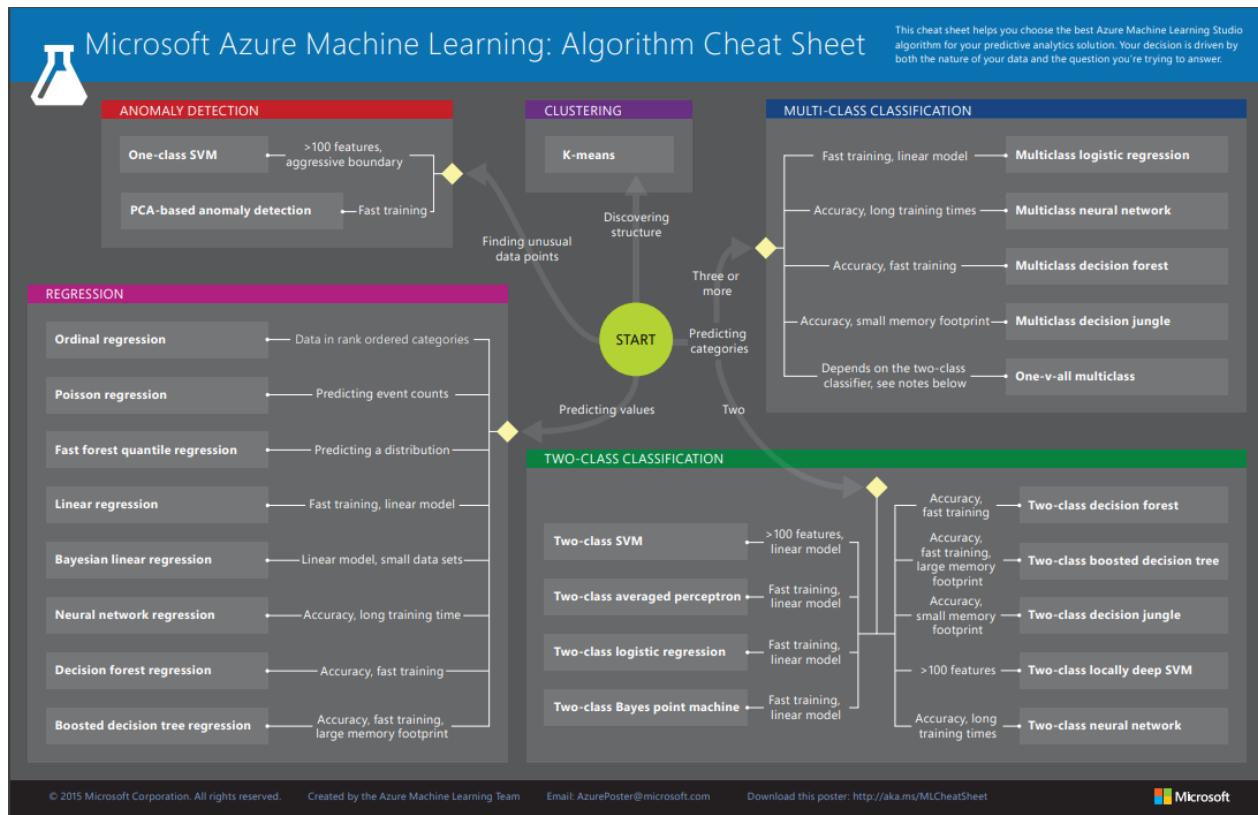
Aqui podem estar algumas ideias para pensar nos modelos a escolher:

- O poder preditivo do modelo
- O tempo de execução do modelo e como ele será dimensionado para muito mais dados
- A interpretabilidade do modelo
- Com que frequência precisaremos executar o modelo e / ou se ele suporta o aprendizado online.
- Distribuição da variável target
- Dados não lineares?
- Outliers?
- Dados faltando?

Também pode verificar este fluxograma do Sklearn:



E este do Azure:



Esta referência fornece muitos recursos de visualização para algoritmos de aprendizado de máquina. Eu recomendo para obter uma intuição visual das técnicas de ML.

Esta referência do sklearn compara os limites de decisão dos principais classificadores.

Em geral, com a seleção de modelos, é uma boa ideia testar métodos mais simples, como Regressão Logística ou Naive Bayes, como referência, e depois passar para classificadores não lineares, como a escolha de Árvores de Decisão. Em projetos reais de produção, a regra de ouro é que você só quer usar a sofisticação que for necessária para maximizar a eficiência.

Student successfully implements a pipeline in code that will train and predict on the supervised learning algorithm given.

Excelente trabalho implementando a função `train_predict!` Esta parte é o principal driver para todo o projeto. Com esse pipeline, você pode ver como o desempenho dos três modelos é alterado ao usar tamanhos de data sets diferentes.

Pipelines ML são ferramentas incríveis. Um Pipeline pode ser especificado como uma sequência de estágios que executa essa seqüência de etapas, uma após a outra. Imagine que você esteja trabalhando em grandes projetos de ML e precise descobrir qual modelo tem o melhor desempenho. Você experimentará os modelos com frequência para ver qual deles funciona melhor e também qual combinação de parâmetros funciona melhor. Um pipeline nos ajuda com tarefas repetitivas sem as mesmas etapas de implementação para cada algoritmo a cada vez. Basicamente ajuda você com o trabalho duro.

O Scikit-learn fornece um utilitário Pipeline para ajudar a automatizar os fluxos de trabalho de aprendizado de máquina. Os pipelines funcionam permitindo que uma sequência linear de transformações de dados seja encadeada, culminando em um processo de modelagem que pode ser avaliado.

Você pode aprender mais sobre [Pipelines em scikit-learn](#).

Student correctly implements three supervised learning models and produces a performance visualization.

Bom trabalho definindo o `random_state`!

Para mais informações sobre por que usamos `random_state`, por favor, dê uma olhada [nesta resposta StackOverflow](#).

Improving Results

Justification is provided for which model appears to be the best to use given computational cost, model performance, and the characteristics of the data.

Boa justificativa para sua escolha em seu modelo.

Pode não ser justo comparar modelos diferentes com seus parâmetros padrão. Por exemplo, as árvores de decisão tendem a não ter um bom desempenho com seus parâmetros padrão, enquanto seu modelo ajustado pode superar o SVM ajustado.

Student is able to clearly and concisely describe how the optimal model works in layman's terms to someone who is not familiar with machine learning nor has a technical background.

Boa explicação. Porém, é necessário ir além. Aqui estão alguns links para ajudá-lo a explicar seu modelo:

- <https://rayli.net/blog/data/top-10-data-mining-algorithms-in-plain-english/>
- <http://blog.echen.me/2011/03/14/laymans-introduction-to-random-forests/>
- <https://prateekvjoshi.com/2014/05/05/what-is-adaboost/>
- <http://xgboost.readthedocs.io/en/latest/model.html>

The final model chosen is correctly tuned using grid search with at least one parameter using at least three settings. If the model does not need any parameter tuning it is explicitly stated with reasonable justification.

Ótimo uso do GridSearch aqui com todas as combinações e estados aleatórios.

Dica Pro: Com um conjunto de dados desequilibrado como este, uma ideia para garantir que os labels sejam divididos igualmente entre os conjuntos de validação seria usar o StratifiedShuffleSplit do sklearn: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

```
from sklearn.model_selection import StratifiedShuffleSplit  
cv = StratifiedShuffleSplit(...)  
grid_obj = GridSearchCV(elf, parameters, scoring=scorer, cv=cv)
```

Student reports the accuracy and F1 score of the optimized, unoptimized, models correctly in the table provided. Student compares the final model results to previous results obtained.

Ótimo resultado. Por, aqui é necessário responder as perguntas:

Estes scores são melhores ou piores do que o modelo antes da otimização?
Como os resultados do modelo otimizado se comparam aos benchmarks do naive predictor que você encontrou na Questão 1?_

Feature Importance

Student ranks five features which they believe to be the most relevant for predicting an individual's' income. Discussion is provided for why these features were chosen.

Muito intuitivo. A seleção das features é uma etapa muito importante no fluxo de trabalho de qualquer algoritmo de aprendizado de máquina. Principais razões para usar a seleção de recursos são:

- Permite que o algoritmo treine mais rápido
- Reduz a complexidade de um modelo e facilita a interpretação
- Melhora a precisão de um modelo se o subconjunto correto for escolhido

Student correctly implements a supervised learning model that makes use of the `feature_importances_` attribute. Additionally, student discusses the differences or similarities between the features they considered relevant and the reported relevant features.

É difícil de adivinhar.

Você pode perceber que a maioria dos recursos 'importantes' são recursos numéricos, alguma idéia de por que isto ocorre?

Não há muitos algoritmos que usam `feature_importances_` (alguns usam `coef_`, mas não são intercambiáveis).

Student analyzes the final model's performance when only the top 5 features are used and compares this performance to the optimized model from Question 5.

O objetivo desta seção é destacar o fato de que nem sempre precisamos necessariamente usar todas as features.

Dica: Em vez de escolher apenas um subconjunto de features, seria uma boa ideia testar algoritmos como [PCA](#). Você vai estudar isso no próximo projeto!

 REENVIAR

 BAIXAR PROJETO



Melhores práticas para sua resubmissão do projeto

Ben compartilha 5 dicas úteis para a revisão ressubmissão do seu projeto.

[▶ Assistir Vídeo \(3:01\)](#)

[RETORNAR](#)

Avalie esta revisão

[FAQ do Estudante](#)