Machine Learning Engineer Nanodegree

Projeto Capstone

Gerson Felipe Schwinn

02 de Novembro de 2017

I. Definição

Visão Geral do Projeto

A diversificação de investimentos, quando bem feita, é uma alternativa para melhorar a rentabilidade financeira a médio e longo prazo. Ela é importante na construção de um patrimônio, e deve ser elaborada de acordo com os objetivos do investidor [1].

Uma das formas de diversificação de investimentos são os imóveis. Há uma série de vantagens conhecidas como a segurança, potencial de valorização e proteção contra a inflação. Suas principais desvantagens são a falta de liquidez e custos gerados como manutenção, impostos, contas e condomínio enquanto o imóvel estiver em seu poder [2].

O objetivo deste projeto é a criação de uma ferramenta automatizada que irá coletar dados de imóveis a venda nos principais sites de imobiliárias definidas pelo autor e filtrar deste conjunto os imóveis que tendem a ter potencial de valorização, chamados aqui como um “bom investimento”.

O público alvo são investidores que não necessariamente são do ramo imobiliário, mas pessoas que pretendem diversificar seus investimentos aplicando em imóveis com boa chance de retorno financeiro e que não dispõem de muito tempo disponível para garimpar boas oportunidades de negócios.

Declaração do problema

Uma das formas para encontrar uma boa oportunidade de investimento em imóveis é através da visitação recorrente dos sites das imobiliárias e análise individual de cada imóvel a venda. Porém esta é uma tarefa que pode ser cansativa e demorada quando se há uma grande quantidade de ofertas à disposição.

O principal problema a ser solucionado é dispensar do investidor a necessidade de análise individual de uma grande quantidade de imóveis disponíveis à venda, trazendo a ele somente registros com potencial para investimento, economizando assim o seu tempo.

Além da economia de tempo proporcionada, existe a possibilidade de se encontrar boas oportunidades em menos tempo, diminuindo assim a concorrência pela aquisição do mesmo por outros investidores.

Para conseguir atingir este objetivo, propõem-se a criação de um robô que irá navegar nos sites das imobiliárias e extrair imóveis a venda de forma automatizada, exportando estes dados no formato CSV que posteriormente irá ser consumido por um programa que irá filtrar os potenciais imóveis através do uso de *machine learning*.

Métricas

A métrica para avaliação do programa será a taxa de acerto de reconhecimento/classificação de imóveis que possuam potencial para investimento.

O programa será treinado e testado com um conjunto de dados onde há informações dos imóveis como valor, localização, quantidade de cômodos, metragem, etc. e um rótulo definindo se este registro é considerado um bom investimento ou não.

II. Análise

Exploração de dados

Os dados necessários para a criação deste projeto foram extraídos do site de uma das maiores imobiliárias da cidade de Santa Cruz do Sul / Rio Grande do Sul. Para limitar a quantidade de dados, foi definido que a avaliação será feita sobre imóveis do tipo “Casa”.

Os imóveis disponíveis podem ser vistos no seguinte link: http://www.imoveiscatedral.com.br/imoveis/a-venda/casa/santa-cruz-do-sul

Os dados não se apresentam numa forma normalizada. É necessário realizar a visitação da página e extrair dados relevantes. Como um dos objetivos do projeto é que a coleta de dados seja feita de forma automática, foram avaliadas ferramentas para acessar a página do site da imobiliária e extrair suas informações. A ferramenta escolhida foi a de nome *puppeteer* [3].

Puppeteer é uma biblioteca nodejs que fornece uma API de alto nível para controlar o navegador *Headless* Google Chrome sobre o protocolo *DevTools*.

O arquivo resultante dessa exploração do site da imobiliária é um arquivo .csv que possui as seguintes informações:

- Valor do imóvel, em formato decimal
- Bairro, em formato texto
- Observações, em formato texto
- Tipo, em formato texto
- Quantidade de dormitórios, numérico
- Quantidade de banheiros, numérico
- Vagas na Garagem, formato numérico
- Metragem, formato decimal
- Código do imóvel, formato texto
- Bom Investimento, Booleana

Exemplo:

```
900000.00;Ana Nery;;Casa em Santa Cruz do Sul;3;;;256.45;CA0198-CA1Q;0

900000.00;Country;Condomínio 450;Casa em Santa Cruz do
Sul;3;;;260.68;CA0304-CA1Q;0

944000.00;Jardim Europa;Condomínio 470;Casa em Santa Cruz do
Sul;3;;;204.74;CA0139-CA1Q;0

954000.00;Jardim Europa;;Casa em Santa Cruz do Sul;3;;1;204.74;CA0230-CA1Q;0

980000.00;Jardim Europa;;Casa em Santa Cruz do Sul;3;;;220;CA0263-CA1Q;0

990000.00;Centro;;Casa em Santa Cruz do Sul;4;;;170;CA0300-CA1Q;0

1000000.00;Higienópolis;;Casa em Santa Cruz do Sul;4;;;500;CA0283-CA1Q;1
```

Existem dados que não são relevantes para a resolução deste problema, como a coluna de observações. Ela é um exemplo de coluna que pode ser removida.

Há colunas que nem sempre estão preenchidas, como Quantidade de Dormitórios, Quantidade de Banheiros e Vagas na Garagem. A falta destes dados pode ocasionar falsos positivos, uma vez que eles são dados relevantes para a avaliação correta de um imóvel.

Existem no conjunto de dados também alguns imóveis onde o preço de venda não foi divulgado, impossibilitando de fazer sua avaliação.

A coluna que identifica um bairro é uma coluna categórica, que poderia ser abordada de outra forma. Ela poderia ser trocada por uma ou mais colunas do tipo Verdadeiro Ou Falso para identificar se aquele bairro é nobre ou não, e/ou se possui colégios próximos, se o índice de violência é baixo ou não. Enfim, informações sobre o bairro que podem ser relevantes para a avaliação do imóvel, caso o resultado obtido não seja satisfatório.

A última coluna, do bom investimento, foi preenchida de forma manual para cada um dos imóveis. Ela foi preenchida com a ajuda de um especialista imobiliário que com sua experiência ajudou a determinar quais eram imóveis que poderiam ser escolhidos por um investidor.

Visualização exploratória

Alguns dados interessantes sobre o dataset são:

Tipo Investimento	Nº Registros	Média Tamanho em m²	Média dormitórios
Bom	17	261,93	2,88
Mal	232	134,50	2,63
Total	**249**	**143,20**	**2,65**

Uma característica que ficou evidente é que o tamanho médio e quantidade média de dormitórios das casas consideradas um bom investimento são maiores que a média do conjunto. Pode-se chegar a conclusão que imóveis que mais dão retorno financeiro são imóveis maiores.

Algoritmos e Técnicas

Os dados para realizar a classificação dos imóveis se encontram em sites da internet das imobiliárias da cidade. Para extrair esses dados para um formato que seja fácil manipular e limpar, será criado um robô usando a ferramenta *Puppeteer* [3], que é uma biblioteca *nodejs* que permite manipular o *google chrome* com uma api amigável, possibilitando assim extrair os dados que estão em páginas html e exportar para um arquivo csv.

Com o arquivo csv em mãos, o mesmo irá ser classificado por um especialista na área de investimentos de imóveis, que irá avaliar cada um dos registros e informar se este é um possível bom investimento ou não. Este arquivo será usado para treinamento e testes.

Uma vez que o resultado desejado neste projeto é identificar se um imóvel possivelmente é um bom ou mal investimento, serão usados alguns algoritmos de classificação disponíveis no *scikit-learn*.

Os algoritmos a serem avaliados serão:

- *DecisionTreeClassifier*
    - Vantagens
        - É um algoritmo simples para entender e interpretar.
        - Pode ser combinado com outras técnicas de decisão
        - Custo computacional baixo
        - Não precisa de muita preparação para os dados
    - Desvantagens
        - Pode ocorrer a criação de árvores muito complexas que podem não lidar bem com generalizações, o chamado overfiting.
        - Ela pode ser instável, uma vez que pequenas variações nos dados pode resultar em uma saída completamente diferente.
        - Existem conceitos difíceis de aprender porque as árvores de decisão não as expressam facilmente, como XOR, paridade ou problemas de multiplexadores.

- *SVM*
    - Vantagens
        - É rápido em conjuntos de dados com muitas dimensões (colunas)
        - Usa um subconjunto de pontos de treinamento na função de decisão (chamado de vetores de suporte), por isso também é eficiente em memória.
        - Versátil: diferentes funções do Kernel podem ser especificadas para a função de decisão. São fornecidos kernels comuns, mas também é possível especificar kernels personalizados.
    - Desvantagens
        - Se o número de recursos for muito maior do que o número de amostras, evitar o excesso de ajuste na escolha das funções do Kernel e o termo de regularização é crucial. Não é o caso para este conjunto de dados. O número de amostras é maior que o de recursos.
        - Falta de transparência nos resultados. Os *SVM's* não podem representar a pontuação de todas as empresas como uma função paramétrica simples dos índices financeiros, pois sua dimensão pode ser muito alta.

Benchmark

Para comparar o desempenho obtido pela minha aplicação, usarei os dois algoritmos citados acima para fazer o treinamento e teste do aplicativo. Os resultados serão comparados e será adotado o algoritmo que atingir uma meta de pelo menos 80% de acertos. Os acertos aqui podem ser definidos como a correta classificação do imóvel.

Para fazer este benchmark, os algoritmos serão treinados com dados reais, e em seguida testados com exemplares de imóveis que não estavam no dataset de treinamento para avaliar seu desempenho e *overfiting*.

Neste momento, não consigo vislumbrar problemas que possam ocorrer na esfera de tempo de processamento, uma vez que a base de dados proposta para este trabalho e sua aplicação não é muito grande. Serão avaliados, numa estimativa inicial, no máximo 1.000 imóveis. Além disso, a taxa de inclusão de novos registros será baixa, não há um grande fluxo de imóveis novos disponíveis para venda por dia.

Logo, a única preocupação para o benchmark no momento é a taxa de sucesso na classificação.

III. Metodologia

Preprocessamento de dados

Uma das etapas para que o produto final seja útil, é a coleta das ofertas de vendas de imóveis a partir de sites de imobiliárias. Para tal, será usado um robô criado por mim com auxílio da biblioteca *Puppeteer.* Este, irá navegar no site informado, e coletará informações relevantes do objeto de pesquisa.

Certos imóveis possuem informações incompletas, como número de cômodos, garagens ou valor comercial. Num primeiro momento serão descartados os registros incompletos, uma vez que com poucas informações disponíveis, ficará muito difícil prever com mais exatidão a viabilidade financeira do negócio.

Pensando em um produto a longo prazo, estes dados faltantes poderiam ser extraídos de sites de outras imobiliárias, uma vez que um mesmo imóvel poderá ser disponibilizado por vários vendedores. Porém, para que isso aconteça, é necessário aplicar novamente a inteligência artificial para conseguir identificar similaridades entre o mesmo imóvel em sites distintos, uma vez que cada imobiliária gera um código identificador e fotos próprias do imóvel. Poderia-se utilizar machine learning para identificar se o imóvel de uma foto é o mesmo que está em outra. Para este trabalho de

conclusão, não será criada essa ferramenta a fim de limitar o escopo de trabalho e ter um produto funcional em menos tempo possível.

A exclusão de ofertas de vendas que não possuem dados suficientes para serem analisados já serão excluídos pelo próprio robô, evitando assim uma carga desnecessária de para a etapa seguinte.

Um exemplo de imóvel com dados suficientes para avaliação pode ser visto na figura 1.

Figura 1:

Este imóvel possui valor de venda, seguido do bairro onde está localizado, a cidade, número de dormitórios, banheiros, vagas na garagem, tamanho e código de identificação.

Certos imóveis não continham a informação de número de banheiros, para estes foi usado o valor padrão 1. O campo vagas na garagem também faltava em certos registros, quando este não existia foi assumido o valor 0 para esta coluna. No caso da coluna dormitórios, quando não existia, o valor padrão passou a ser 1, pois seguindo a mesma lógica do banheiro, toda casa tende a ter um quarto e banheiro. Imóveis que não tinham informação de tamanho, foi usado valor 0.

Implementação

Primeiramente faço a importação das bibliotecas necessárias para fazer a leitura do arquivo de treinamento e testes, e bibliotecas com os algoritmos para classificação dos registros.

```
#Importação das bibliotecas para leitura do arquivo csv, classes para divisão em treinamento e testes,
#algoritmos DecisionTreeClassifier e svm
import pandas as pd
from IPython.display import display
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
```

Em seguida, carrego o arquivo com os dados para treinamento e testes.

```
#Leitura do arquivo para treinamento e testes
try:
    data = pd.read_csv("imoveis-treinamento-testes.csv", sep=";")
    print "Arquivo de treinamento e testes de imóveis carregado com sucesso!"
except:
    print "Dataset não pode ser carregado!"
```

Aqui faço um tratamentos das colunas mais importantes e são removidos registros que possuem preço “sob consulta”.

```
#Troca 0.0 para False e 1.0 para True na coluna bomIvestimento
data['bomInvestimento'] = data['bomInvestimento'].map({0.0: False, 1.0: True})

#Preenche valores padrões para coluna de banheiros, vagas na garagem, dormitórios
#E metragem da casa
data['banheiros'] = data['banheiros'].fillna(1)
data['vagasGaragem'] = data['vagasGaragem'].fillna(0)
data['dormitorios'] = data['dormitorios'].fillna(1)
data['metragem'] = data['metragem'].fillna(0)

#Remove imóveis com preço sob consulta
data = data.drop(data[data.valor == "Sob consulta"].index)
```

Em seguida, são exibidas algumas informações e estatísticas úteis do *dataset* .

```
#Mostro os tipos de dados de cada coluna para saber se a conversão foi feita corretamente
print data.dtypes

#Mostro algumas estatísticas do dataset
display(data.describe())

#Outras estatísticas sobre o conteúdo do dataset
print "Bons investimentos: " + str(len(data[data.bomInvestimento]))
print "Maus investimentos: " + str(len(data[data.bomInvestimento == False]))
print "Tamanho médio das casas com bom investimento: " + str(data[data.bomInvestimento].metragem.mean())
print "Tamanho médio das casas mal investimento: " + str(data[data.bomInvestimento == False].metragem.mean())
print "Número médio de dormitórios bom investimento: " + str(data[data.bomInvestimento].dormitorios.mean())
print "Número médio de dormitórios mal investimento: " + str(data[data.bomInvestimento == False].dormitorios.mean())
```

Nesta parte, faço a remoção das colunas que no momento, não tem relação com a classificação do imóvel. Removi também a coluna *bomInvestimento* que é a classificação dos imóveis feitas pelo especialista na área.

```
# Fiz uma cópia do dataframe original removendo colunas não usadas para o fim de treinamento e testes
new_data = data.drop(['bomInvestimento', 'codigo', 'bairro', 'obs', 'tipo'], axis = 1)
```

Nesta parte é realizada a divisão dos dados, 30% para treinamento e 70% para testes.

```
#Divisão do dataset para fins de treinamento e testes
X_train, X_test, y_train, y_test = train_test_split(new_data, data['bomInvestimento'], test_size=0.30, random_state=158)
```

Abaixo aplico o algoritmo de árvore de decisão para o treinamento e em seguida o teste. É informada em seguida a taxa de acertos.

```
#Aplicando o algoritmo DecisionTreeClassifier e exibindo o score
dt = DecisionTreeClassifier(random_state=160)
dt.fit(X_train, y_train)
print "Score usando algoritmo DecisionTreeClassifier: " + str(dt.score(X_test, y_test))
```

O score obtido com essa classificação foi de 0.906666666667

Em seguida, apliquei o algoritmo SVC para comparar os resultados.

```
aSvm = svm.SVC(random_state=160)
aSvm.fit(X_train, y_train)
print "Score usando algoritmo SVC: " + str(aSvm.score(X_test, y_test))
```

Neste algoritmo, sem realizar nenhum refinamento ou ajuste fino, foi alcançado um score de 0.946666666667

Refinamento

Para o refinamento dos resultados, usei o algoritmo *GridSearchCV* que testa exaustivamente parâmetros do algoritmo de classificação desejado, para tentar

encontrar a melhor combinação de parâmetros possíveis, aumentando assim a precisão da classificação.

No código abaixo, fiz testes com parâmetros *criterion, random_state* e *min_samples_split* do algoritmo *Decision Tree*.

```
#Usando GridSearchCV para encontrar parâmetros melhores para os algoritmos de classificação
parametersDT = {
    'criterion': ['gini', 'entropy'],
    'random_state': range(1,50),
    'min_samples_split': [2,3]
}
regressorDT = DecisionTreeClassifier()
clf2 = GridSearchCV(regressorDT, parametersDT, n_jobs=2)
clf2.fit(X_train, y_train)
score = clf2.best_estimator_.score(X_test, y_test)
print "Score do algoritmo DecisionTreeClassifier com GridSearchCV: " + str(score)
print "Melhores parâmetros encontrados para DecisionTreeClassifier"
print clf2.best_params_
```

O score alcançado foi de 0.96 , superior ao resultado anterior, sem os ajustes. No resultado anterior obtivemos score de 0.906666666667

No código abaixo, fiz ajustes para o algoritmo *SVC*.

```
parametersSVC = {
    'C': [0.01, 0.1, 0.5, 1] ,
    'kernel': ['sigmoid', 'rbf'],
    'random_state': range(1,100)
}
regressorSVC = svm.SVC()
clf1 = GridSearchCV(regressorSVC, parametersSVC, n_jobs=2)
clf1.fit(X_train, y_train)
# TODO: Reportar a pontuação da previsão utilizando o conjunto de teste
score = clf1.best_estimator_.score(X_test, y_test)
print "Score do algoritmo SVC com GridSearchCV: " + str(score)
print "Melhores parâmetros encontrados para SVC"
print clf1.best_params_
```

Após executar esse código, não foram encontrados parâmetros que melhorassem a classificação. O score final continuou igual ao original, ou seja, 0.946666666667

IV. Resultados

Avaliação e validação de modelos

Pelos testes realizados, o modelo que saiu com melhor *score* foi *Decision Tree* com uma taxa de acertos de 96%. Além de ser um modelo de bom desempenho computacional, ele obteve ótimos resultados.

Os parâmetros finais após os testes exaustivos usando o algoritmo *GridSearchCV* foram:

```
{'min_samples_split': 2, 'random_state': 3, 'criterion': 'gini'}
```

Justificação

Comparando com o resultado esperado no capítulo *benchmark*, que era de no mínimo 80%, o algoritmo *Decision Tree* é um candidato viável para ser usado em ambiente de produção uma vez que sua taxa de acertos foi de 96%.

Este é um ótimo percentual que pode trazer benefícios ao usuário da aplicação.

V. conclusão

Visualização de formulário livre

Para este trabalho em particular, o objetivo é conseguir uma maior índice possível de precisão na classificação dos imóveis como possíveis candidatos a um “bom investimento”.

Com uma taxa de acerto de 96% pode-se afirmar que o modelo proposto tem potencial para se tornar uma ferramenta útil de auxílio para o investidor.

Claro que é necessário uma análise ainda muito mais completa do imóvel através de visitas e inspeções do bem, porém com este programa é possível diminuir consideravelmente a necessidade de avaliação de muitos imóveis otimizando melhor o tempo livre do investidor.

Reflexão

O primeiro problema a se resolver era a origem dos dados. Não existem arquivos normalizados disponíveis para download. Os dados estão em publicações nas redes sociais e nos sites das imobiliárias em formato html. Esta foi a primeira questão a se resolver.

Eu, no meu papel de projetista de sistemas web, conhecia ferramentas que automatizam a navegação em sites e sua extração de dados. Antes deste projeto, no meu ambiente de trabalho já tive necessidade de usar algumas destas, isso ajudou muito no processo de escolha. Costumo acompanhar os lançamentos de novos projetos pelos trends do github, foi lá que conheci o projeto puppeteer.

Essa ferramenta permite navegar em sites e executar códigos arbitrários no contexto da página, isso possibilita a extração dos dados e exportação no formato CSV ou outro qualquer.

Neste momento eu possuía um arquivo, porém seus registros não estavam classificados como bom ou mal investimento. Para que eu pudesse treinar e testar os algoritmos propostos, foi necessário que tivesse previamente uma análise de cada imóvel para que ele fosse classificado. Essa classificação foi feita avaliando registro a registro, com a ajuda de um especialista na área imobiliária. Ele me forneceu a lista de códigos de imóveis que ele avaliou como bom investimento, analisando principalmente a combinação de fatores como tamanho da casa, quantidade de cômodos, banheiros, garagem e claro, o valor de venda.

Esta coluna booleana, chamada *bomInvestimento*, foi adicionada ao arquivo csv.

Em seguida, foi usada a biblioteca pandas para leitura do arquivo. Colunas não categorizadas que não são necessárias para avaliação do imóvel para este projeto foram excluídas, como código do imóvel, tipo, observações e bairro.

Na sessão de melhorias faço observações sobre o campo tipo do imóvel, que para este projeto não é importante, uma vez que o trabalhos e limita a avaliar somente bens do tipo “casas”, logo optei em excluir ele.

Nos registros que não haviam dados, como número de cômodos, garagem, banheiro e tamanho da casa assumi valores padrões, descritos na seção Pré Processamento para que fosse possível avaliá-los pelos algoritmos propostos.

Neste ponto os dados estavam normalizados, e prontos para a fase de aprendizado e teste. Foi feita a separação dos dados para aprendizado e teste usando o método *train_test_split* disponível na biblioteca sklearn.

Em seguida, foram instanciados objetos das classes dos algoritmos *DecisionTreeClassifier* e *SVC*. Foi realizado o treinamento e teste com cada um deles, e seu resultado foi comparado.

Neste primeiro momento, sem ajustes de parâmetros, o algoritmo SVC teve um score de 0.94666 o que é um valor que já estaria dentro do *benchmark* proposto. Enquanto o modelo Decision Tree teve um score de 0.906666

Após isso, foi usado o algoritmo GridSearchCV para que se tentasse encontrar parâmetros melhores para cada modelo. Com ele foi obtido um resultado melhor no modelo *DecisionTreeClassifier*, enquanto no outro não houve melhoras no score.

Melhoria

Para que o projeto seja atrativo para um público alvo maior, umas das alterações necessárias seria a inclusão e análise de mais tipos de imóveis. Para a realização deste trabalho foram definidos o tipo de imóvel “casa” por ser o mais encontrado na cidade onde realizei a análise.

Porém será necessário adicionar análise para apartamentos, duplex, terrenos, coberturas, chácaras a fim de atender aos anseios de outros usuários.

Outro ponto a ser abordado é a origem de dados. Neste projeto foi utilizado um site de imobiliária para alimentar o programa, porém, é importante como produto, que seja feita uma busca em mais sites, e que seja feito um tratamento especial dos dados para que se evite trazer imóveis repetidos. Este aspecto não foi tratado neste projeto porque os registros do mesmo imóvel em diferentes sites de imobiliárias contém identificações e fotos distintas. Como a localização exata não é fornecida, teria que ser usada uma inteligência artificial que poderia analisar as fotos de cada anúncio de venda e verificar que se trata do mesmo imóvel ou não.

Em paralelo a questão da origem dos dados, pode-se obter mais detalhes de um determinado imóvel para melhorar a eficácia do aplicativo. Determinados sites de imobiliárias disponibilizam mais informações que outros, logo, estes dados podem ser mesclados para fornecer uma quantidade de informação maior.

VI. Referências

[1] https://urbe.me/lab/porque-e-uma-boa-opcao-investir-em-imoveis/

[2] http://www.3ainvestimentos.com.br/2017/03/17/527/

[3] https://github.com/GoogleChrome/puppeteer