* **Introdução**

Bitcoins e outras criptomoedas estão sendo muito exploradas no mundo atualmente e sua variação tem um valor elevado.

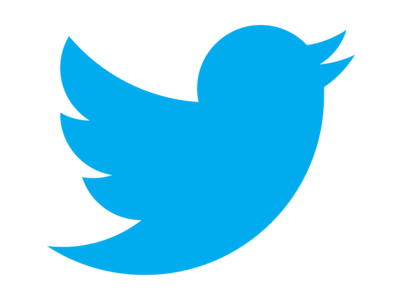
Juntamente com o mundo conectado que vivemos, podemos acompanhar em redes sócias informações on-line sobre bitcoins, informações essas que nos ajuda tomar decisões se está na hora de entrar nesse mercado ou não.

Por esse motivo, me senti motivado a minerar o Twitter em busca uma ajuda para identificar se o que estão falando no momento sobre bitcoin é bom ou ruim e possivelmente utilizar essa informação para acompanhar o mercado de Bitcoins para poder tomar uma decisão de quando será o melhor momento para a compra dessa criptomoeda.

* **Definição do Problema**

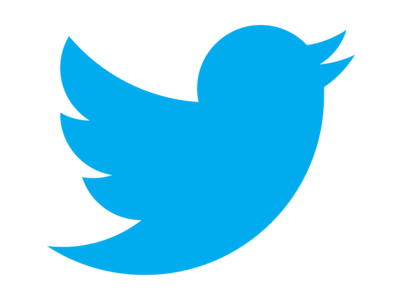
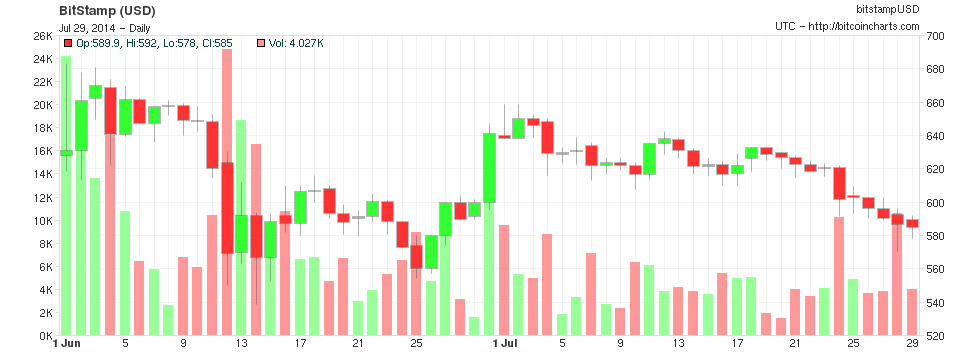
Primeiramente irei capturar em dias diferentes uma quantidade razoável de Tweets, o qual são textos publicados no Twitter, que contenham a palavra bitcoin. Após a captura desses Tweets, irei analisa-los e classifica-los manualmente como Positivo, Negativo ou Neutro. Após essa classificação usarei esses textos (Tweets) como entrada para um modelo de classificação.





Treinamento e Teste

Tweets com a palavra Bitcoin Classificados previamente



Previsão

Tweets Novos

Correlação entre a classificação prevista e o valor do Bitcoin

* **Conjunto de dados**

Para realização desse projeto usarei os dados coletados do Twitter, no caso aproximadamente 5000 Tweets.

As informações necessárias para isso são:

* + **USR**
    - Nome do usuário do Twitter que publicou esse Tweet
  + **TWEET**
    - O texto publicado pelo usuário, o qual sempre terá a palavra bitcoin.
  + **LANG**
    - Linguagem em que o Tweet foi publicado
  + **DATE**
    - Data de publicação do Tweet
  + **CLAS**
    - Classificação que eu darei para esse Tweet, sendo: Positiva, para as publicações que me influenciariam a comprar bitcoin: Negativa, para as publicações que me influenciariam a não comprar bitcoin: Neutra, para as publicações que não tem influência na compra de bitcoin.

Para captura dos dados do Twitter utilizei um código em Python criado por mim, mas usando como referência:

<https://apps.twitter.com/app/14711331/keys>

<https://ronanlopes.me/coletor-de-tweets-em-python-com-o-tweepy/>

* **Solução do Problema**

Após realizar a classificação dos dados, irei utilizar esses textos (Tweets) como entrada para alguns modelos de classificação e testar qual modelo de classificação traz o melhor F1-Score para prever a mesma classificação dada anteriormente.

Para isso irei tirar algumas métricas como estatística descritiva, verificar a necessidade da criação de um Bag of Words (com stemming e stop words por exemplo), estudar a possibilidade de gerar um word embeddings por word2vec, depois executar 2 ou 3 modelos de classificação que são aconselhados para text mining, como por exemplo Guassian Naive Bayes, Multinomial Naive Bayes, SVM e Random Forest em cima de 70-80% dos Tweets (texto utilizado como dado de entrada no modelo), e por fiz realizar avaliações e validações sobre esse modelo, como por exemplo usar um Cross-Validation e também Grid Search para melhor ajustar meu modelo.

“Você mencionou a utilização de cross-validation (na solução do problema). Visto que seus dados são gerados ao longo do tempo, tome bastante cuidado ao usar essa forma de analisar. Pense assim, se a validação cruzada (tradicional) vai fazer um shuffle dos seus dados, então você vai usar registros gerados no passado e no futuro para treinar um modelo que vai ser testado em dados do futuro e do passado. Na prática, você vai poder apenas treinar com dados do passado e testar com aqueles gerados no futuro, certo? Então a avaliação mais realista seria aquela em que você faz uma validação cruzada levando isso em consideração, ou seja, os folds são organizados conforme a data em que foram gerados (só treina com dados do passado para testar com dados do futuro). Dê uma olhada na TimeSeriesSplit da sklearn para ver como isso pode ser automatizado.”

* **Métricas de Avaliação e validação com um Benchmark**

A métrica que utilizarei para avaliação do modelo será o F1\_Score, pois assim

terei uma avaliação tanto de precisão quando de recall do modelo.

Por fim, mesmo após avaliar e validar meu modelo com os dados de teste, irei capturar dados novos, prever sua classificação com meu modelo, verificar a porcentagem de positivos e negativos e compara-la com o verdadeiro valor do bitcoin para aquele dia e o dia seguinte, o que seria um Benchmark.

Dessa forma posso assumir que meu modelo acertou ao me ajudar tomar uma decisão de compra de Bitcoin. Nesse caso, posso assumir que sempre que a previsão for positiva e o valor do bitcoin aumentou consideravelmente (valor em % e a ser definido) meu modelo acertou no auxilio da tomada de decisão de compra do Bitcoin.

Como benchmark para validar a sanidade do meu modelo final, irei utilizar uma classificação aleatória, por exemplo o [DummyClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html) do sklearn, afim de medir se meu modelo tem resultados melhores ou piores que a classificação aleatória. Em caso de obter classificações piores que o benchmark terei um bom indicio de que algo não foi realizado corretamente no meu modelo, e como plano de contorno poderei treinar com outro algoritmo.

* **Fluxo de Trabalho**

Esse trabalho será desenvolvido seguindo os seguintes passos e as seguintes ferramentas:

1. **Aquisição dos dados**: primeiro vou obter os dados usando a API do Twitter, tweepy;
2. **Rotulação**: será realizada uma rotulação dos dados manual de acordo com a influência do Tweet para compra de bitcoins. Sendo: Positiva, para as publicações que me influenciariam a comprar bitcoin: Negativa, para as publicações que me influenciariam a não comprar bitcoin: Neutra, para as publicações que não tem influência na compra de bitcoin.
3. **Preparação dos dados:** irei verificar a necessidade de criar uma Bag of Words ou um word2vec (<https://machinelearningmastery.com/develop-word-embeddings-python-gensim/>) para melhor treinar o modelo, utilizando bibliotecas do Python.
4. **Separar os dados:** os dados serão separados entre treino e teste, buscando utilizar entre 70-80% dos dados para treino e o restante para teste. Para isso irei utilizar a biblioteca sklearn do Python.
5. **Treinar 3 modelos:** irei utilizar os dados de treinamento separados anteriormente para treinar 3 modelos diferentes utilizando a biblioteca sklearn do Python.
6. **Escolher o melhor modelo:** dentre os 3 modelos treinados anteriormente, irei escolher o que tiver um melhor F1\_Score, fornecido pelo sklearn.
7. **Melhorar o modelo escolhido:** com o modelo já escolhido, irei utilizar um Grid Search com Cross Validation, sklearn, para otimizar o F1\_Score do modelo, passando os parâmetros necessários para o modelo escolhido.
8. **Treinar modelo Benchmark:** utilizarei os mesmos dados de treino para treinar um modelo de classificação aleatória, [DummyClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html).
9. **Comparar resultados:** irei comparar o F1\_Score do modelo final e do modelo benchmark. Caso o benchmark seja melhor que o modelo final, irei voltar ao passo 6 e escolher outro modelo.
10. **Analisar resultados:** irei comparar os resultados obtidos com meu modelo e os valores dos bitcoins para o mesmo período, afim de verificar se o modelo prevê Tweets positivos quando o valor do bitcoin sobe e Tweets negativos quando o valor do bitcoin cai.