**Introdução**

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

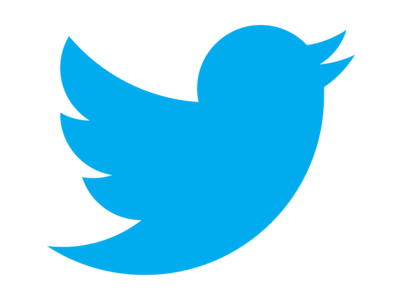
Juntamente com o mundo conectado que vivemos, podemos acompanhar em redes sócias informações on-line sobre bitcoins, informações essas que nos ajuda tomar decisões se está na hora de entrar nesse mercado ou não.

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**

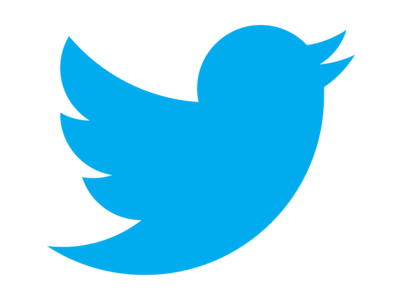
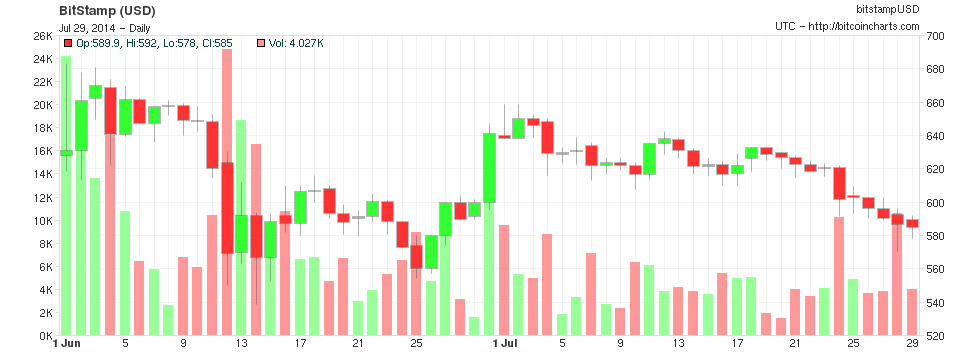
Primeiramente capturei em dias diferentes uma quantidade razoável de Tweets, o qual são textos publicados no Twitter, que contenham a palavra bitcoin. Após a captura desses Tweets, analisei e classifiquei eles manualmente como Positivo, Negativo ou Neutro. Após essa classificação usei esses textos (Tweets) 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**

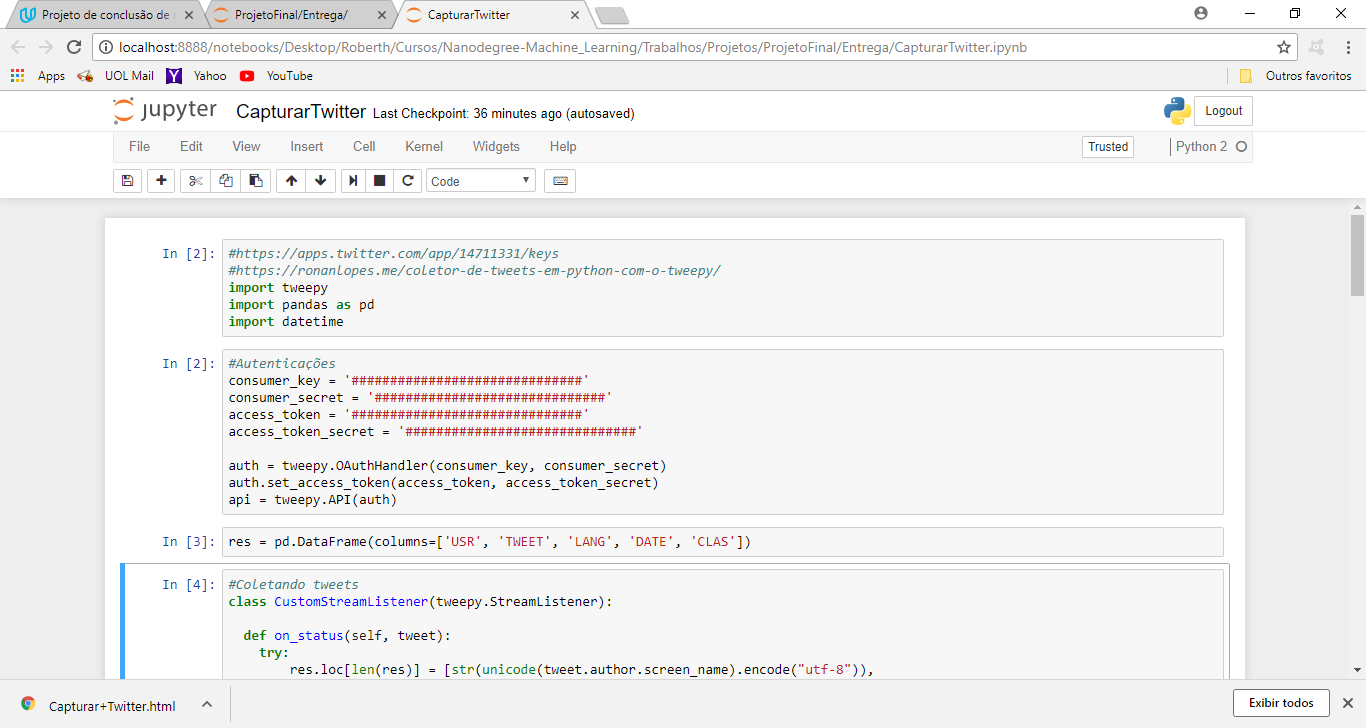
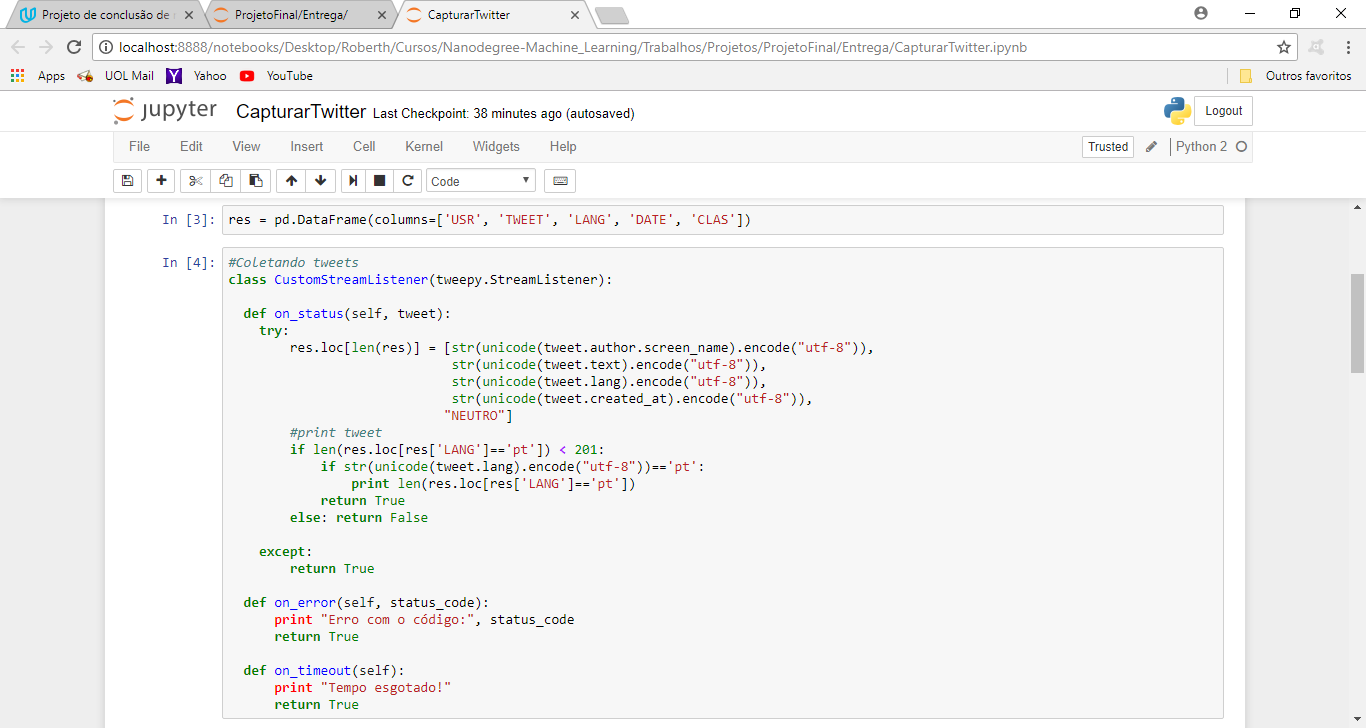
Para realização desse projeto usei os dados coletados do Twitter, no caso aproximadamente 270.000 Tweets.

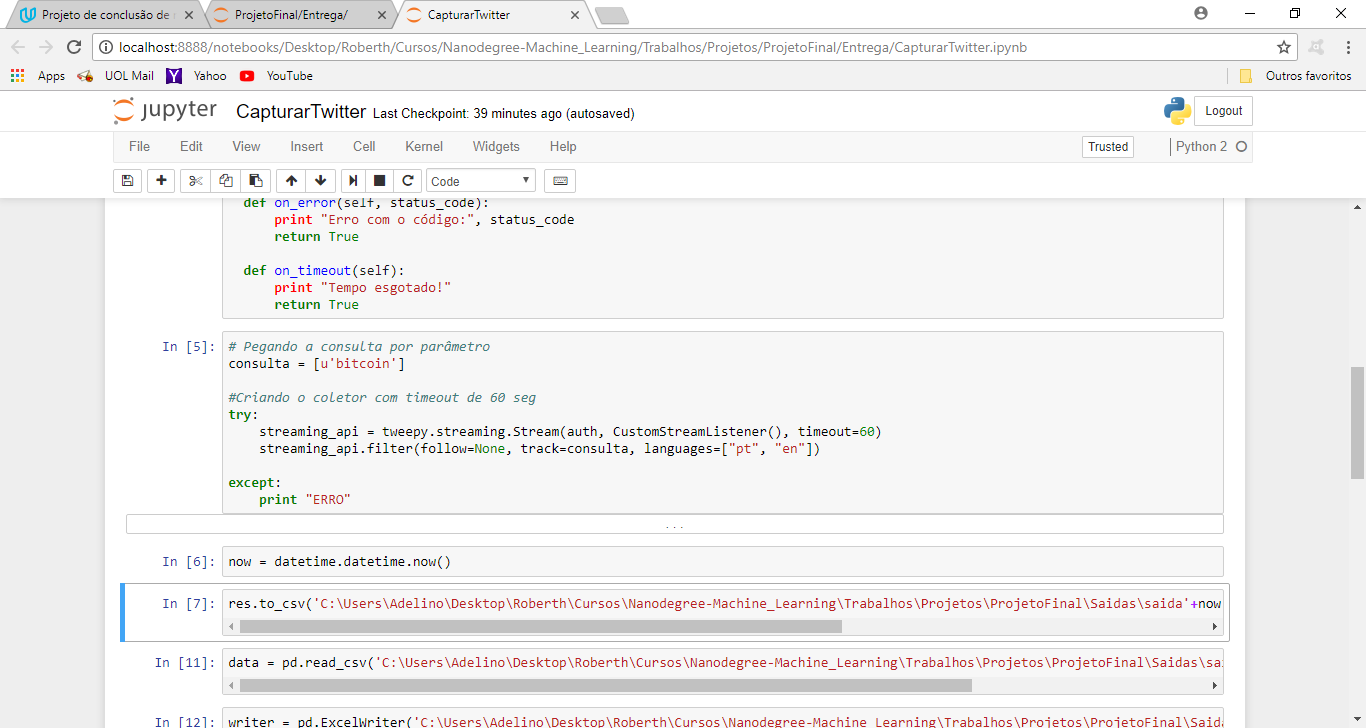
As informações coletadas foram:

* + **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**
    - No ato da captura dos tweets todos vieram como Neutro, mas em seguida eu classifiquei alguns como veremos na seção seguinte, sendo: Positiva, para as publicações que me influenciariam a comprar bitcoin: Negativa, para as publicações que me influenciariam a não comprar bitcoin: Neutro, 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:

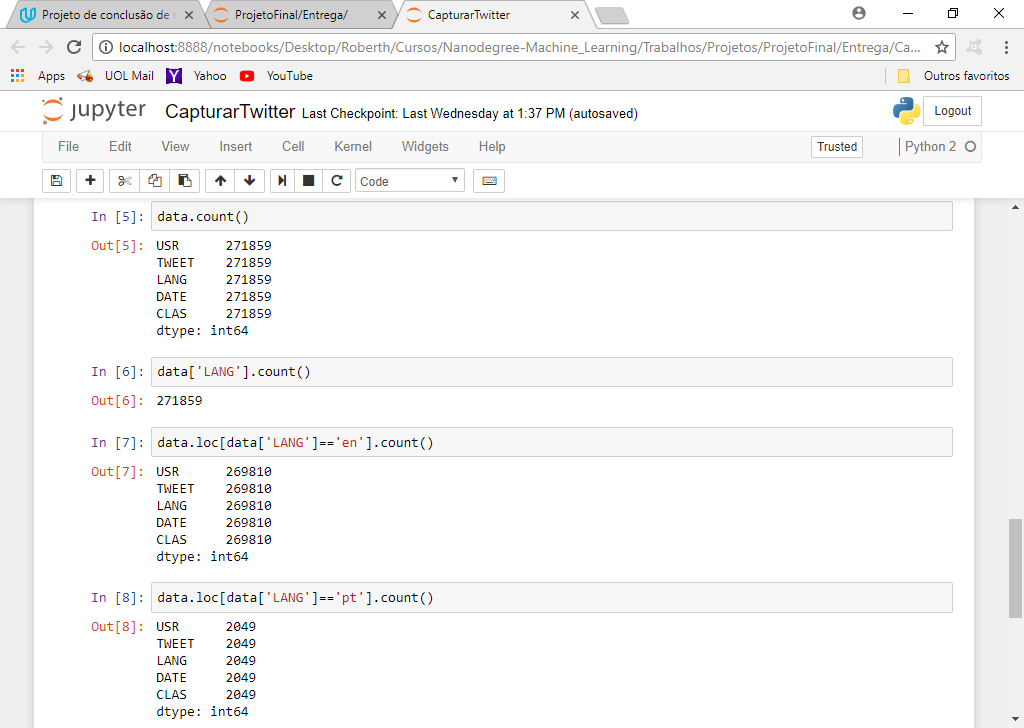
<https://apps.twitter.com/app/14711331/keys>

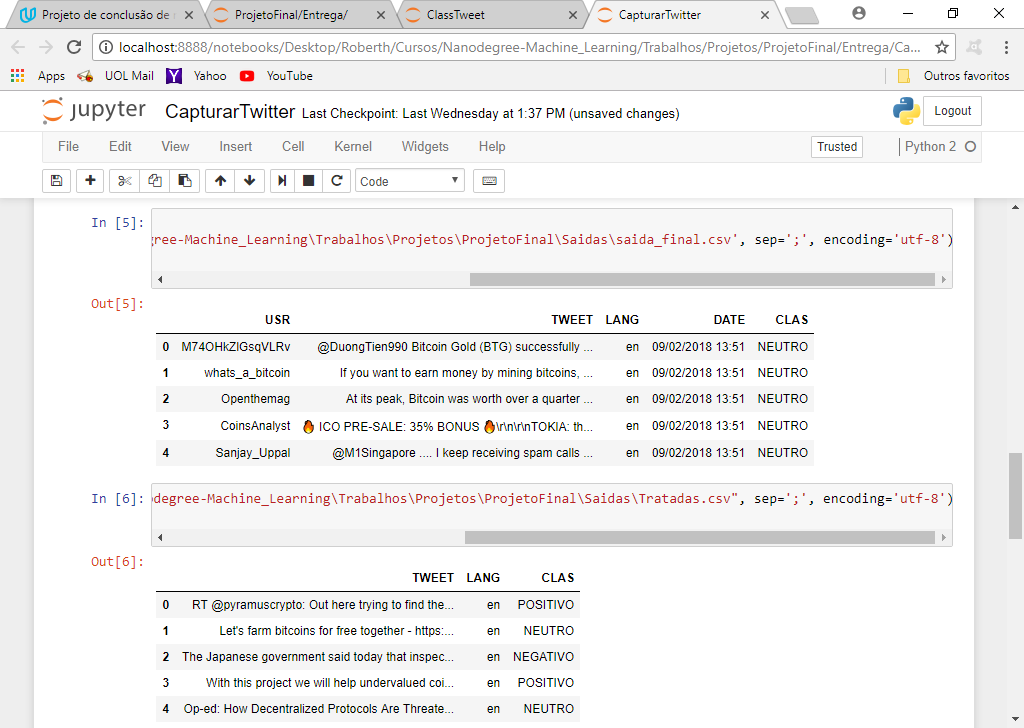
<https://ronanlopes.me/coletor-de-tweets-em-python-com-o-tweepy/>



Exploração de dados

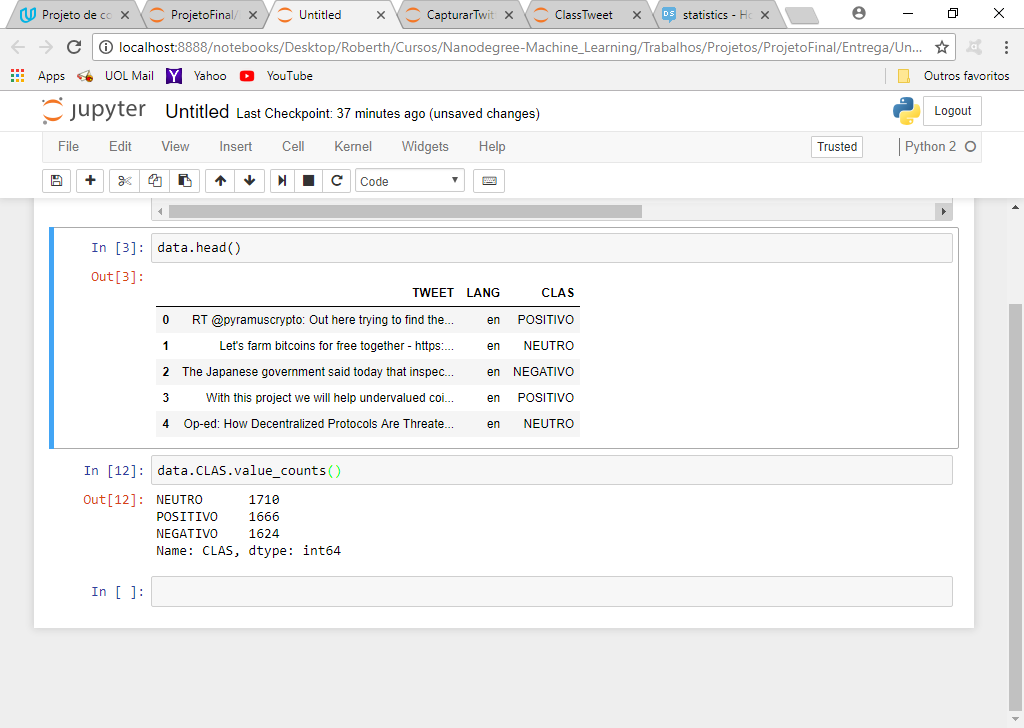
Com do tweets adquiridos passei para a analise do conteúdo. Conforme imagem abaixo, menos de 1% dos tweets eram do idioma português, então decidi abandar os tweets em português e manter apenas os que estavam na língua inglesa.



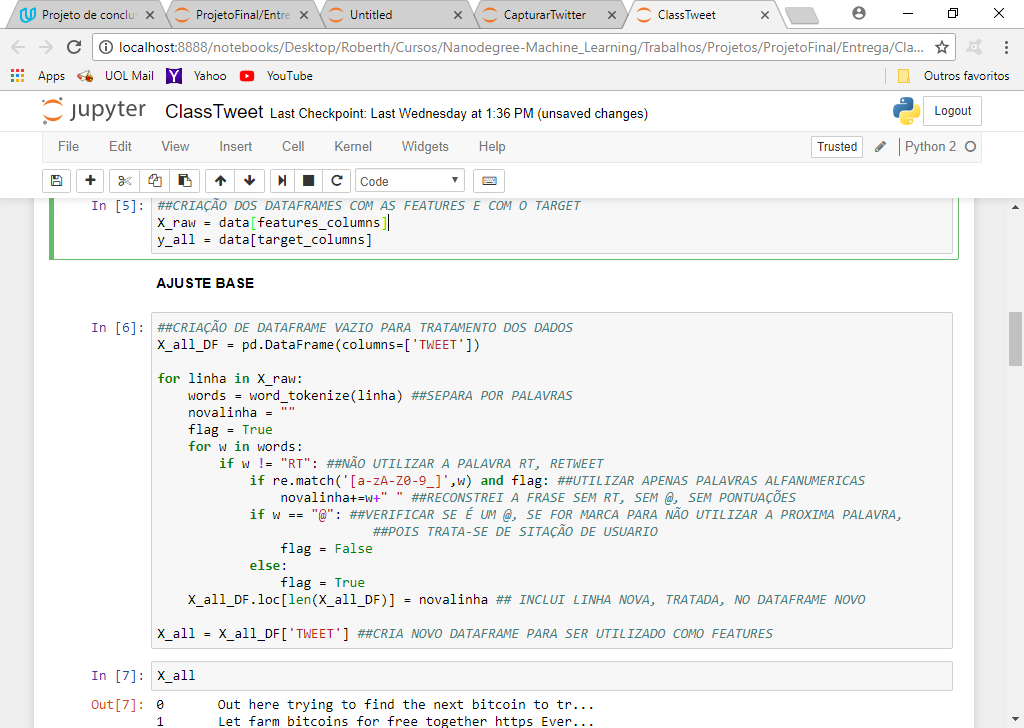
Restando apenas os tweets em inglês passei para a fase de classificação manual dos tweets, classificando-os como Positivo, Negativo e Neutro conforme visto na seção anterior. Entretanto, classificar aproximadamente 200.000 tweets não era viável, então ordenei os tweets de forma aleatória, abandonando assim a data em que o tweet foi postado. Nessa limpeza já removi o usuário que escreveu o tweet, pois para esse projeto o usuário não era necessário. Feito isso fiquei com uma base com apenas tweets em inglês contendo apenas as colunas TWEET, LANG (que não foi utilizada) e CLAS, como podem ser vistas na imagem abaixo:

Após essa alteração no arquivo dos tweets iniciei minha classificação manual seguindo os critérios preestabelecidos, porém após mais de 15.000 tweets classificados identifiquei que a maioria era neutro, tendo uma proporção de 0,6% Negativos, 7% Positivos e 92,4% Neutros. Então decidi continuar a classificação buscando apenas os tweets Negativos e Positivos para chegar em um dataset equilibrado entre os Positivos, Negativos e Neutros.

Feito isso cheguei em novo dataset com 5.000 tweets classificados conforme abaixo:



Dessa forma tenho um dataset balanceado para poder iniciar o processo de criação de uma modelo, porém, antes do inicio da modelagem identifiquei que muitos tweets possuíam palavras que não deveriam ser analisadas, como “RT” (retweet), “emoji”, “@” (antecede o nome de um usuário). Para esses casos eu criei o código abaixo para fazer uma limpeza do texto dos tweets.



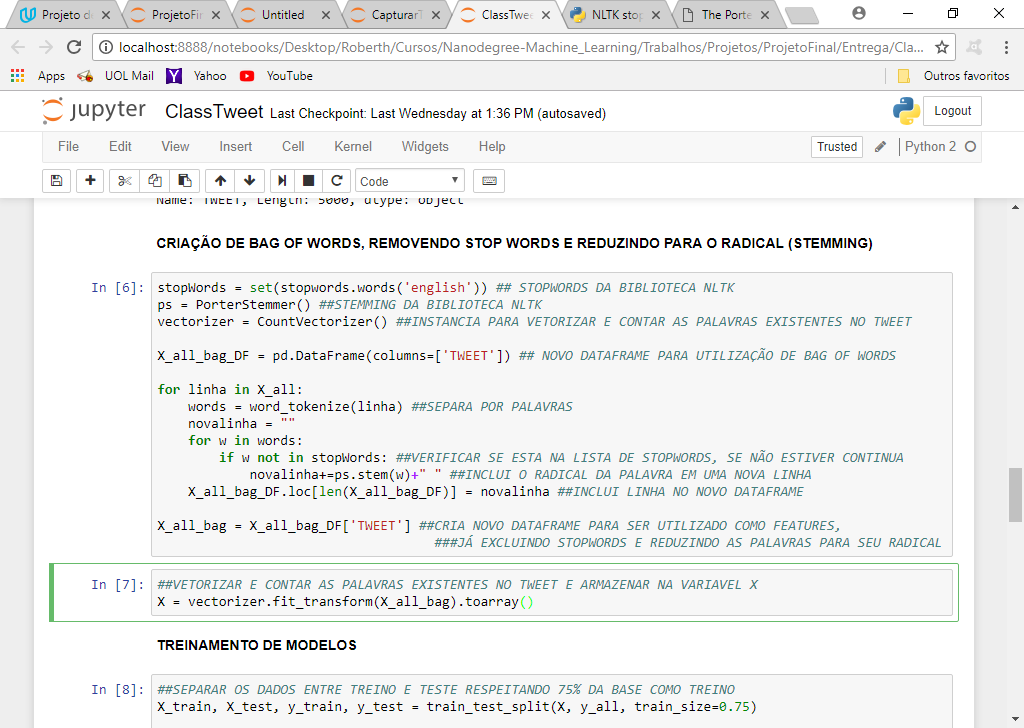
Algoritmos e Técnicas

Pela natureza desse projeto, onde gostaria de identificar o sentimento de cada tweet mencionando a palavra bitcoin, algumas técnicas recomendadas são Bag of Words, remoção de Stop Words, Stemming, criação de um Corpus e Word2Vec.

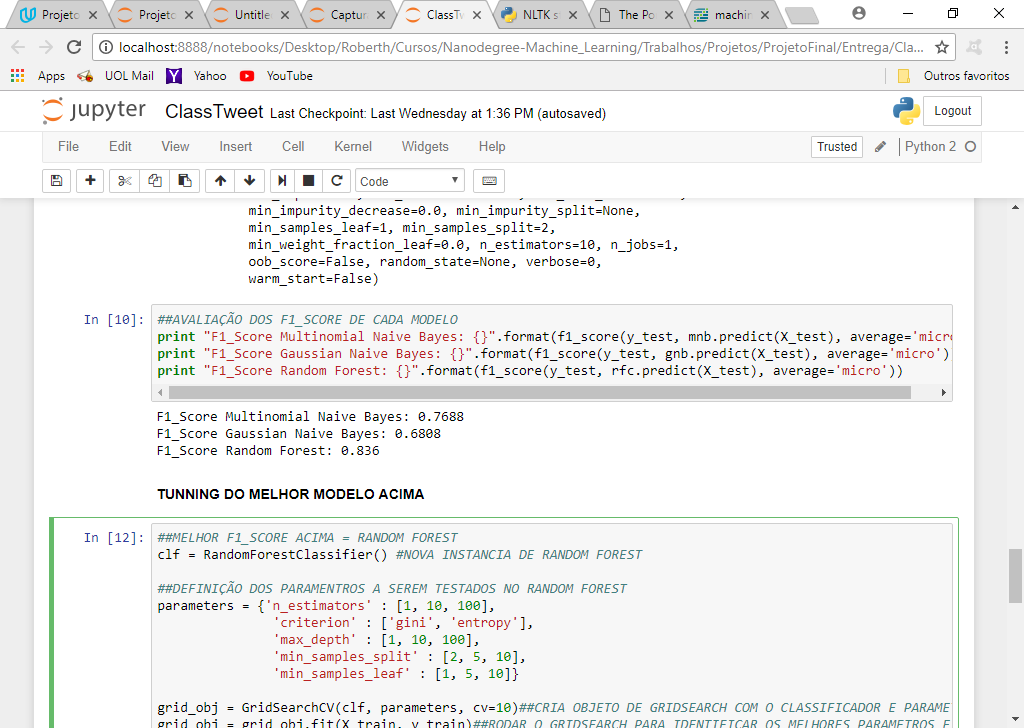
Decidi utilizar a criação de bag of words, onde conteve todas as palavras da minha base de treino vetorizadas para analise de quantidade de palavras em cada tweet. Além de considerar o set de stopwords da biblioteca NLTK (<https://pythonspot.com/nltk-stop-words/>) para filtrar palavras que não possuem peso para a analise de texto, como: the, is, are....ente outras palavras da língua inglesa.

Além de considerar stop words, decidi utilizar a técnica de Stemming, também da biblioteca NLTK (<http://www.nltk.org/howto/stem.html>), para reduzir as palavras para seu radical. Essa técnica permite trabalhar com palavras que possuem o mesmo radical como se fossem a mesma palavra, dessa forma podemos dar mais peso para as palavras, pois estamos considerando todas as formas que essa palavra possa ter aparecido como apenas uma forma.

Para vetorização, remoção de stop words e stemming das palavras eu criei o código abaixo:

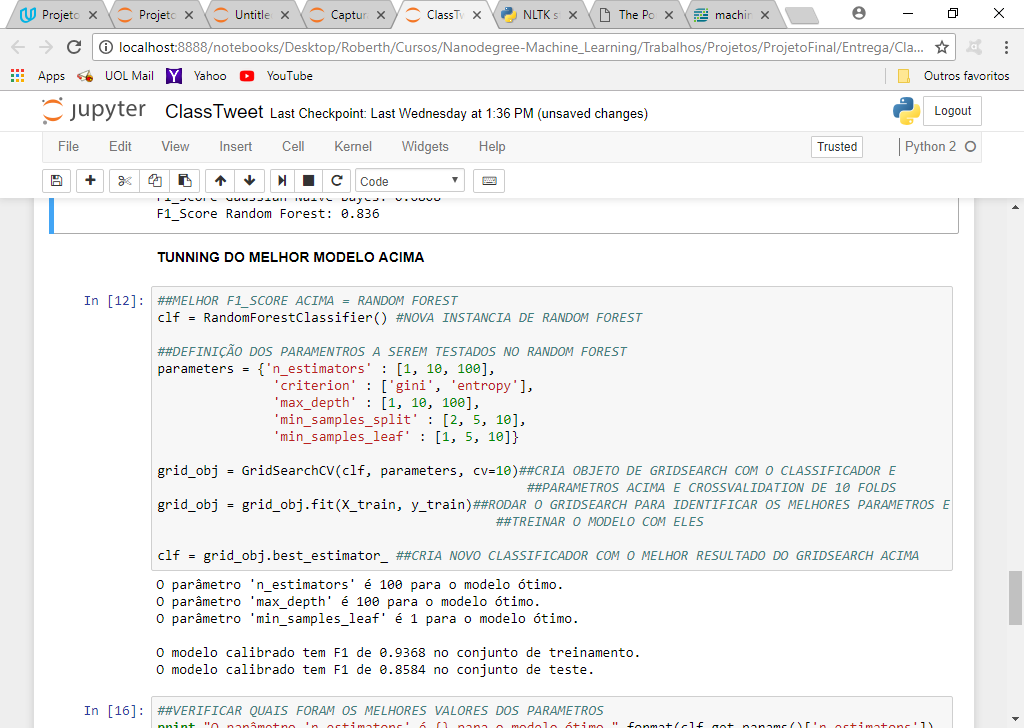


Com o dataframe pronto, vetorizado, utilizei um holdout com 75% da base para treino e os outros 25% para teste. Separados dessa forma apliquei 3 algoritmos que são indicados para analise de texto, 2 tipos diferentes de Naive Bayes pois ele trabalha bem com analise de texto exatamente por considerar que cada palavra é um componente independente, além de um Random Forest, pois assim como arvore de decisão o RF é um ótimo classificador.

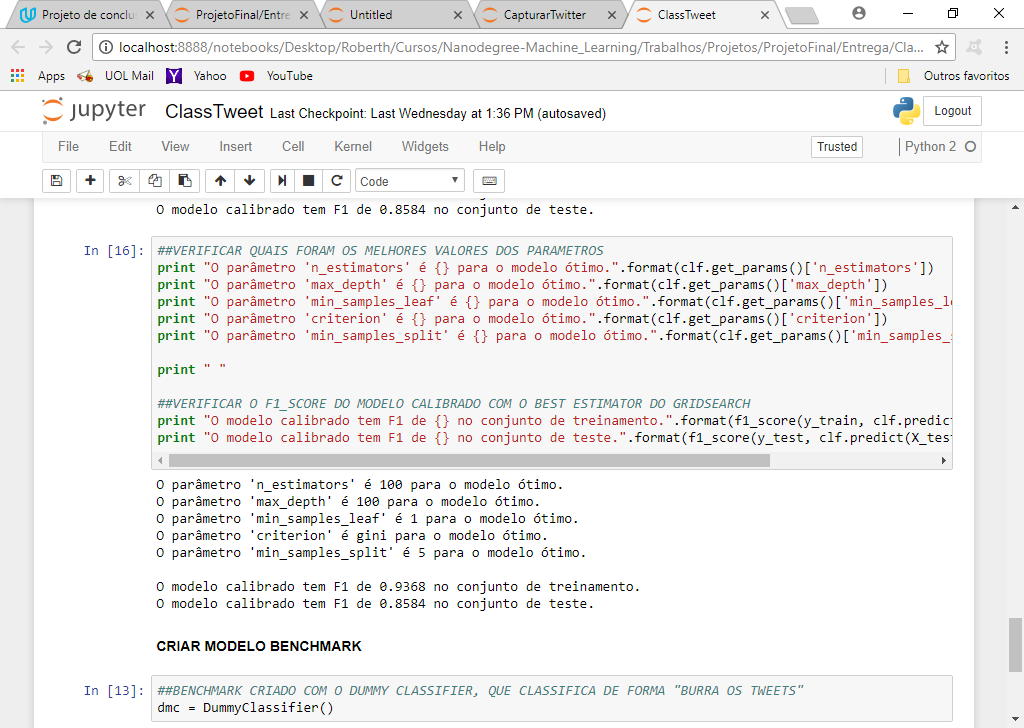
Para analisar qual dos modelos teve um melhor desempenho utilizei o F1\_Score, pois assim consigo mensurar a precisão e o recall do modelo em um indicador único. Tendo como resultado para o melhor modelo o Random Forest, conforme abaixo:

Dado o melhor modelo, considerando o F1\_Score dos dados de teste, decidi tentar melhorar o modelo buscando um melhor F1\_Score atrás do Grid Search, o qual executa o modelo alterando os parâmetros previamente definidos buscando quais são os parâmetros necessários para se alcançar o melhor modelo.

Para o Random Forest escolhi os parâmetros “n\_estimators” (numero de arvores na floresta), “criterion” (forma e qualidade de como será dividido os dados nas arvores), “max\_depth” (profundidade máxima de cada arvore), “min\_samples\_split” (número mínimo de amostras em cada separação), “min\_samples\_leaf” (número mínimo de amostrar para compor uma folha da arvore). Abaixo segue codificação:



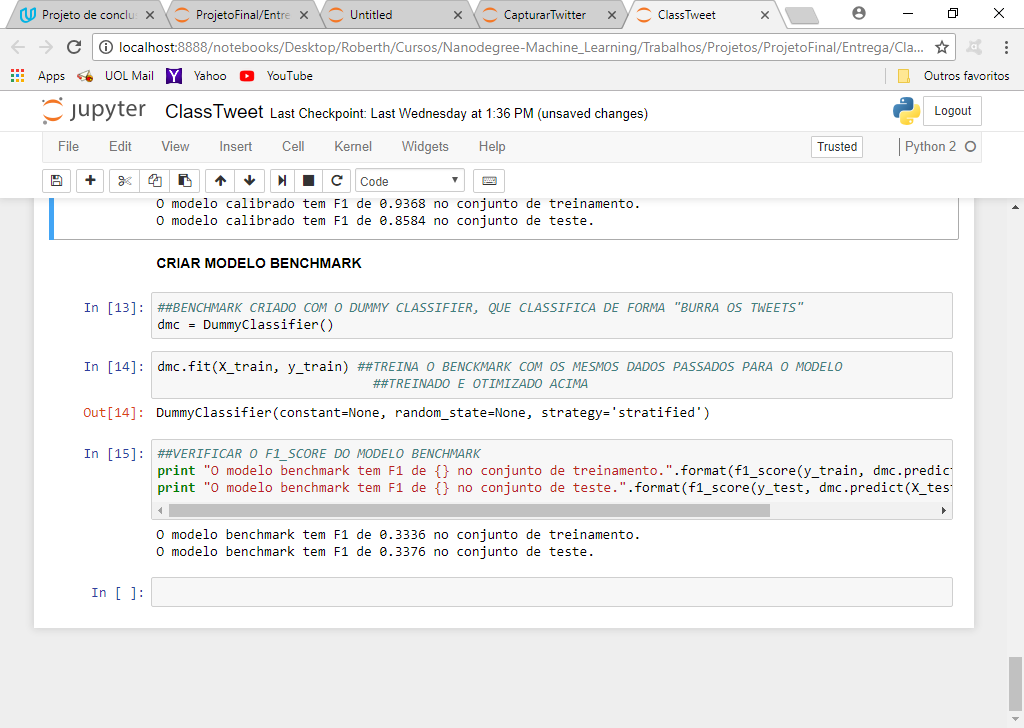
Obtivo os seguintes resultados:



Com esses ajustes obtidos no GridSearch temos um modelo com F1\_Score 2% melhor que o modelo anterior. O ganho não foi muito significante, mas como não houve um aumento considerável de tempo de execução considero que esse modelo seja melhor que o anterior. Podemos notar também que o modelo está fit, pois tanto o treino quando o teste estão com F1\_Score próximos e altos, mostrando que o modelo acerta bem para os dados novos e para os dados de treino.

Modelo de referência

Após o desenvolvimento desse modelo um modelo de benchmark foi criado para aferir melhor a qualidade do meu modelo. Como benchmark para validar a sanidade do meu modelo final, utilizei uma classificação aleatória, o [DummyClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html) do sklearn (<http://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html> ), afim de medir se meu modelo tem resultados melhores ou piores que a classificação aleatória.

 Como podemos ver abaixo, a implementação do modelo benchmark aleatório, os resultados do F1\_Score são bem piores que meu modelo, garantindo assim mais uma vez de que a classificação feita pelo meu modelo está boa.

**Metodologia**

* **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.

Pré-processamento dos dados

Todos os passos de pré-processamento de dados são documentados com clareza. Anormalidades ou características dos dados são ajustadas conforme necessário. Se não for necessário nenhum pré-processamento, isso é justificado detalhadamente.

Todas as etapas de pré-processamento foram claramente documentadas. Anormalidades ou características sobre os dados ou entradas que precisavam ser endereçadas foram corrigidas. Se não for necessário um pré-processamento de dados, foi claramente justificado.

Implementação

O processo de implementação das métricas, algoritmos e técnicas utilizados com o conjunto de dados é detalhadamente documentado. Dificuldades ocorridas durante a implementação são documentadas e discutidas.

O processo para o qual métricas, algoritmos e técnicas foram implementadas com os conjuntos de dados ou dados de entrada dados foi completamente documentado. As complicações ocorridas durante o processo de codificação são discutidas.

Refinamento

O processo de aprimoramento dos algoritmos e técnicas utilizados é claramente documentado. Os resultados iniciais e finais são relatados, além de passos intermediários quando cabível.

O processo de melhoria nos algoritmos e técnicas utilizadas está claramente documentado. As soluções iniciais e finais são relatadas, juntamente com soluções intermediárias, se necessário.

**Resultados**

Avaliação e validação do modelo

O modelo final, incluído seus parâmetros, é avaliado em detalhes. É feita uma análise para validar a robustez da solução encontrada.

As qualidades do modelo final - como os parâmetros - são avaliadas em detalhes. Algum tipo de análise é usado para validar a robustez da solução do modelo.

Justificativa

Os resultados finais são comparados com o modelo de referência escolhido através de algum método estatístico. A conclusão de que o modelo desenvolvido é suficiente para resolver adequadamente o problema é justificada.

Os resultados finais são comparados com o resultado ou o limite de referência com algum tipo de análise estatística. A justificação é feita para saber se o modelo final e a solução são suficientemente significativos para ter resolvido adequadamente o problema.

**Conclusão**

Visualização de forma livre

Foi criada uma visualização enfatizando uma qualidade importante do projeto, acompanhada de discussão detalhada. Indicativos visuais são claramente definidos.

Foi fornecida uma visualização que enfatiza uma qualidade importante sobre o projeto com uma discussão aprofundada. As pistas visuais estão claramente definidas.

Reflexão

A solução do problema de ponta a ponta é resumidamente discutida, destacando aspectos particularmente interessantes ou desafiadores do projeto.

O aluno resuma de forma adequada a solução de problemas de ponta a ponta e discute um ou dois aspectos específicos do projeto que eles acham interessante ou difícil.

Aperfeiçoamento

O relatório discute como pelo menos um aspecto da implementação que possa ser aperfeiçoado. O resultado potencial desse aperfeiçoamento é comparado ou contrastado com a solução atual.

É feita uma discussão sobre como um aspecto da implementação poderia ser melhorado. As soluções potenciais resultantes dessas melhorias são consideradas e comparadas / contrastadas com a solução atual.

**Qualidade**

Apresentação

O relatório do projeto segue uma estrutura bem organizada e é compreensível para sua audiência. Cada seção é escrita de maneira clara, concisa e específica. O texto não contém erros gramaticais ou ortográficos em excesso. Todas as referências utilizadas na confecção do projeto são citadas.  
O relatório do projeto segue uma estrutura bem organizada e será facilmente entendido pelo público que pretende. Cada seção é escrita de forma clara, concisa e específica. Poucos erros gramaticais e gramaticais estão presentes. Todos os recursos utilizados para concluir o projeto são citados e referenciados.

Funcionalidade

O código é bem formatado e contém comentários para explicar implementações completas. O output do código reflete os resultados discutidos no relatório.

O código é formatado cuidadosamente com comentários que efetivamente explicam implementações complexas. A saída produz resultados e soluções semelhantes às discutidas no projeto.