



Relatório do Projeto Aplicado

Nome Cristiano Cesar da Silva Oliveira

Título Classificação de emoções em Tweets
relacionados ao mercado de ações
brasileiro

Curso MBA em Machine Learning

Orientador(a) Msc. Matheus Mendonça

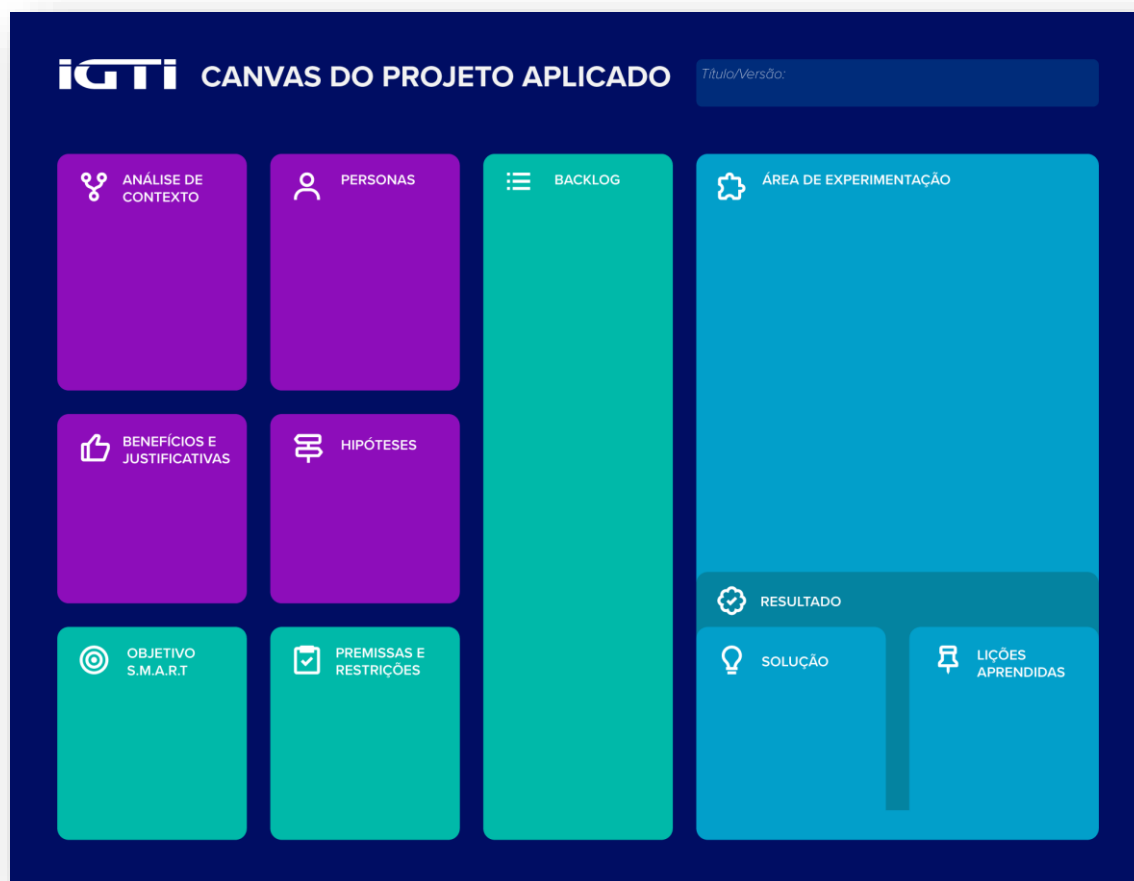
Data 07 de novembro de 2021

Sumário

1. CANVAS do Projeto Aplicado	4
1.1 Desafio	5
1.1.1 Análise de Contexto	5
1.1.2 Personas	8
1.1.3 Benefícios e Justificativas	11
1.1.4 Hipóteses	12
1.1.5 Solução	14
1.1.6 Objetivo SMART	14
1.1.7 Premissas e Restrições	15
1.1.8 Backlog de Produto	16
2. Área de Experimentação	17
2.1 Sprint 1	17
2.1.1 Solução	17
2.1.2 Lições aprendidas	22
2.2 Sprint 2	23
2.2.1 Solução	23
2.2.2 Lições aprendidas	27
2.3 Sprint 3	28
2.3.1 Solução	28
2.3.2 Lições aprendidas	32
2.4 Sprint 4	33
2.4.1 Solução	33
2.4.2 Lições aprendidas	35
2.5 Sprint 5	36
2.5.1 Solução	36
2.5.2 Lições aprendidas	39
2.6 Sprint 6	40
2.6.1 Solução	40
2.6.2 Lições aprendidas	43
2.7 Sprint 7	44
2.7.1 Solução	44
2.7.2. Lições aprendidas	49
3. Considerações Finais	50
3.1 Resultados Finais	50
3.2 Contribuições	52
3.3 Próximos passos	52

1. CANVAS do Projeto Aplicado

Figura conceitual, que representa todas as etapas do Projeto Aplicado.



1.1 Desafio

Desenvolver e implementar um modelo para classificação de emoções em Tweets relacionados ao mercado de ações brasileiro.

Do ponto de vista de negócio, considera-se como cliente em potencial Corretoras de Valores que desejam capturar e compreender a reação de investidores pela emoção expressas em Tweets, de forma a obter informações que possam, porventura, servir como indicadores na definição de estratégias e diretrizes de atuação, bem como, se viável, empregar os resultados do modelo na concepção de produtos e serviços.

Para tanto, o processo de criação de um modelo classificação terá como fonte de dados no seu desenvolvimento um *dataset* denominado “*Brazilian Stock Market Tweets with Emotions*” disponível na plataforma *Kaggle*. Este *dataset* contém Tweets relacionados a ações do IBOVESPA com emoções anotadas por *crowdsourcing*.

Do ponto de vista da aplicação, como objetivo principal e primário, buscar-se-á a construção de uma solução *on premise* pautada na adoção de técnicas de aprendizado de máquina para a criação de um modelo responsável pela classificação da emoção contida em Tweets, o modelo em si será construído com aplicação de técnicas de processamento de linguagem natural e do treinamento de diferentes algoritmos de classificação.

Novos tweets utilizados para avaliar o desempenho do(s) modelo(s) construído(s) serão capturados por uma API integrada ao Twitter.

Almeja-se ainda como objetivo secundário, mas não obrigatório, fazer o *deploy* do modelo, simulando a aplicação deste em um ambiente de produção, de forma o modelo seja integrado a uma Web API, e caso viável e pertinente hospedar a solução em um ambiente *Cloud*.

1.1.1 Análise de Contexto

Em uma fase inicial de imersão preliminar, como forma de modelar o problema e suas causas, visando: explorar melhor o seu contexto, levantar diferentes percepções e responder questões essenciais sobre o projeto, se fez uso da Matriz Certezas, Suposições e Dúvidas (CSD) que permite compreender o problema sobre diferentes óticas, conforme apresentada na tabela a seguir:

Tabela 1 - Matriz CSD

		Matriz CSD - Listar todas as Certezas, Suposições e Dúvidas		
		Certezas	Suposições	Dúvidas
Diferentes Óticas de Análise	Atores	(1) Corretoras desejam saber mais sobre as emoções dos investidores; (2) Parcela de Investidores que utilizam o Twitter expressão emoções sobre o mercado de ação brasileiro.		
	Cenários		Emoções expressas por investidores em comentários no Twitter podem ser usadas como indicadores para embasar estratégias ou até serviços e produtos	Será que as emoções identificadas podem ser utilizadas de forma agregadora na concepção de novas estratégias, ou mesmo produtos e serviços?
	Regras	Anonimato dos usuários dos Tweets analisados		

Ainda analisando o contexto do problema, prosseguindo para uma fase de imersão profunda, de forma a tentar compreender e modelar as aspirações do usuário e alcançar melhores *insights* da solução, foi aplicada a ferramenta de observações POEMS, conforme segue apresentado na tabela a seguir:

Tabela 2 - Observações POEMS

PESSOAS	OBJETOS	AMBIENTE	MENSAGEM	SERVIÇOS
Quem está presente no contexto em análise?	Que objetos fazem parte do ambiente?	Quais são as características do ambiente?	Que mensagens são comunicadas?	Quais serviços são oferecidos?
Corretores, Analistas de dados	Tweets, Modelos de classificação	Rede social, Internet	Texto contendo emoções sobre o resultado de ações	Capacidade de classificação de emoções relacionadas ao mercado de ações
Investidores		Corretora de Valores		Insights para definição de indicadores estratégias a partir dos resultados obtidos

1.1.2 Personas

Como forma de melhor entender as características dos usuários da solução que será desenvolvida foi criado um biotipo para a sua persona conforme é apresentado na figura a seguir:

Figura 1 – Persona do usuário da solução a ser desenvolvida



José
Investidor

Empresa: Corretora de Valores
Idade: 35 anos
Genêro: Masculino
Educação: Ensino superior
Mídias: Lê a revista Exame e usa ativamente o twitter
Objetivos: Quer viver da valorização e dos rendimentos dos seus investimentos.
Desafios: Lidar com a expectativa relacionada a variação do mercado.
Como minha empresa pode ajudá-la: Identificado comportamentos e sugerindo serviços e produtos.

 **rockcontent**
ResultadosDigitais

Ainda como forma de entender melhor as características dos usuários e seus anseios quanto ao problema a ser solucionado foi proposto também a criação de um Mapa de Empatia cujas perguntas pertinentes e as respostas alcançadas seguem apresentadas adiante.

Mapa de Empatia

1. Com quem estamos sendo empáticos? Corretoras que buscam entender o comportamento de potenciais investidores brasileiros.

Quem é a pessoa que queremos conhecer? Corretoras que querem novas ferramentas que auxiliam no processo de tomada de decisão.

Em que situação (ambiente) ela está? Escritórios, Internet, Twitter, Rede Social.

Qual o papel dela nesta situação? Sujeito que busca realizar previsões de investidores que expressam opinião contendo emoções sobre a variação de ações no mercado financeiro.

2. O que ela precisa fazer? Modelo de classificação de emoções em tweets.

O que deve ser diferente? Ser uma tarefa automatizada.

Quais decisões ela precisa tomar? Avaliar os resultados alcançados e interpretá-los para tomada de decisão.

Como saberemos que a atividade é bem-sucedida? Pelo grau de melhoria alcançado através dos resultados obtidos pelo modelo em um cenário que demanda esta classe de informações para decisão.

3. O que ele vê? A possibilidade de extrair informações de redes sociais no emprego de uma ferramenta que pode agregar valor ao processo de tomada de decisões.

Como é o ambiente no qual está inserido? Mercado de ações.

O que os outros estão falando e fazendo? Não há informações sobre soluções neste seguimento adotadas pelos concorrentes.

O que está lendo e assistindo? Sobre a expansão do Big Data e as potencialidades da aplicação de inteligência artificial.

Que tipo de problemas ele enxerga no dia a dia? Pouca ou nenhuma informação sobre as emoções de investidores quanto a volatilidade das ações no mercado.

4. O que ele Fala? Que precisa melhor compreender as emoções dos investidores.

O que já escutamos ele falando? Que existem algoritmos que classificam emoções através de um texto.

O que imaginamos ele falando? Que aplicações de IA podem incrementar o arcabouço de informações à disposição.

Do que ele reclama? Que carece de mais informações para aperfeiçoar seu conhecimento estratégico.

5. O que ele faz? Contrata uma consultoria e delega a ideia de uma solução para equipe de desenvolvimento.

O que ele faz hoje em dia? Não analisa possíveis emoções sobre o mercado de ações.

Qual o seu comportamento diante de desafios e do sucesso? Busca implementar novas soluções assim como esta, de tentar aplicar IA.

O que imaginamos ele fazendo? Tentando acompanhar as tendências tecnologias disponíveis.

6. O que ele escuta? O Corpo de executivos e conselheiros.

O que as pessoas em seu entorno dizem? Redes sociais tem grande potencial de geração de informações que podem ser utilizadas com propósito estratégico.

O que ele escuta de forma direta e indireta (por meio de outra pessoa)? Que precisa de mais informações tomar decisões mais eficientes.

Quem o influencia? O comportamento dos concorrentes.

Onde ele procura informação? Junto ao departamento de tecnologia da informação.

Quais mídias e ferramentas ele tem acesso? Computadores e internet.

7. O que ele Pensa e Sente?

Dores:

O que ele valoriza e o motiva? Ser vanguarda e referência no seu contexto de negócio.

Quais os sentimentos que o movem? Serem pioneiros no segmento.

Quais são suas expectativas, sonhos e aspirações? Obter informações rápidas e confiável que agreguem valor.

Que desafios enfrenta? Desconhecimento de processo de captação e transformação em massa de dados em informações.

Quais são suas frustrações? Incapacidade de compreender melhor as emoções de investidores.

Quais são os seus medos? Ficarem desatualizados, consequentemente se tornarem irrelevantes e obsoletos.

Desejos:

O que é sucesso para ele? Ser pioneiro e inovador no segmento de mercado atuante.

Qual é a sua ambição? Ser referência.

Onde ele quer chegar? Liderar o segmento em que atua.

O que ele faz para alcançar os objetivos? Contrata profissionais com conhecimento técnico para a construção de soluções que acompanham o estado da arte tecnológico.

Quais são suas vontades e anseios? Melhores resultados.

1.1.3 Benefícios e Justificativas

Uma vez compreendido o problema em seu cenário, tona-se necessário elencar quais tipos de benefícios justificam o prosseguimento deste projeto, para tanto, visando elucidar outros pontos pertinentes se adotou a aplicação da metodologia Blueprint como forma de explorar melhor o problema, a tabela a seguir caracteriza a aplicação do Blueprint para o problema em questão, em sequência segue apresentado o Canvas da Proposta de Valor aplicado como forma de delimitar a solução do projeto.

Tabela 3 Blueprint do projeto aplicado

Ações do Cliente	Classificar emoções expressas em Tweets relacionados a ações e mercado de valores
Objetivos	Extrair informações para enriquecimento estratégico, bem como poder gerar insights na concepção de novos produtos ou serviços
Atividades	Aplicar técnicas Inteligência artificial para capturar emoções em Tweets e posteriormente classificá-las de forma automatizada
Questões	Como capturar Tweets, tratá-los, e classificar as emoções neste contidas de forma automatizada?

CANVAS de Proposta de Valor.

Tarefas do Cliente: Tentar compreender como Tweets pode se relacionar e contribuir ao seu modelo de negócio.

Dores: Necessitar de mais de informações para a tomada de decisão.

Ganhos: Compreender as emoções de investidores e potenciais investidores, bem como possibilitar maior clareza em um cenário de tomada de decisões.

Produtos e Serviços: Modelo de classificação de emoções em Tweets sobre ações no mercado brasileiro.

Analgésicos: Ajudar na compreensão de emoções expressas por investidores.

Criadores de ganho: Diferencial estratégico frente aos concorrentes.

1.1.4 Hipóteses

Em continuidade ao conhecimento adquirido pela análise do problema, pelo aprofundamento do contexto do desafio e pela definição das personas, seguiu-se com o levantamento de hipóteses relacionadas conforme apresentadas na tabela a seguir:

Tabela 4 - Matriz de observações para hipóteses.

Exemplo de observação	Exemplo de Hipótese
Investidores comentam sobre a volatilidade dos valores de ações no Twitter	Investidores querem expressar suas emoções quanto a volatilidade dessas ações no mercado
Corretoras querem entender o que investidores comentam sobre ações	Previsão de emoções em Tweets sobre ações pode ser um indicador para tomada de decisão
Existe textos/palavras que denotam emoções.	Emoções contidas em textos/palavras podem ser classificadas por algoritmos de aprendizado de máquina

Uma vez observadas as hipóteses relacionadas ao contexto do problema, torna-se possível a tarefa de priorizar ideias que apontam e priorizam formas de como solucioná-las.

Possíveis soluções:

- S1 – Pesquisar manualmente Tweets sobre ações do mercado e classificar as respectivas soluções expressadas.
- S2 – Automatizar a classificação de emoções em Tweets sobre ações do mercado por meio de uma solução de inteligência artificial.
- S3 – Realizar somente a captura automática de Tweets, mas classificá-las de forma manual.

Uma vez levantadas as hipóteses (ideias), podem se aplicá-las e respectivamente pontuá-las de acordo com sua prioridade, conforme seguem apresentadas na tabela a seguir:

Tabela 5 - Matriz de Priorização de Ideias.

Matriz Básica							
Soluções	B – Benefícios	A – Abrangência	S – Satisfação	I – Investimentos	C – Clientes	O – Operacionalidade	Total
S1	1	1	1	5	2	1	11
S2	4	5	4	2	3	4	22
S3	3	3	2	4	4	2	18

A figura a seguir, extraída do livro texto da disciplina de Inovação e Design Thinking, contém as métricas aplicadas a matriz de priorização de ideias e foram aplicadas sobre as propostas de soluções sugeridas.

Figura 2 – Balizadores aplicados a matriz de prioridade

Balizadores para notas da Matriz BASICO.						
Escala	B - Benefícios	A - Abrangência	S - Satisfação	I - Investimentos	C - Cliente	O - Operacionalidade
5	De vital importância	Total (de 70 a 100%)	Muito grande	Pouquíssimo investimento	Nenhum impacto	Muito fácil
4	Significativo	Muito grande (de 40 a 70%)	Grande	Algum investimento	Impacto pequeno	Fácil
3	Razoável	Razoável (de 20 a 40%)	Média	Médio investimento	Médio impacto	Média facilidade
2	Poucos benefícios	Pequena (de 5 a 20%)	Pequena	Alto investimento	Impacto grande	Difícil
1	Algum benefício	Muito pequena	Quase não é notada	Altíssimo investimento	Impacto muito grande no cliente	Muito difícil

Fonte: <https://engenhariaexercicios.com.br/gestao-de-qualidade/matriz-qut-basico-conceito-aplicacao-das-matrizes-priorizacao/>.

1.1.5 Solução

Uma vez aplicada sobre o problema a matriz de prioridade de ideias obteve-se a solução S2 como sendo a mais relevante para a resolução do problema, esta que consiste em:

S2 - Utilizar de técnicas de processamento de linguagem natural em conjunto com outros algoritmos de aprendizado de máquina na construção de um modelo de classificação de emoções em Tweets sobre ações do mercado de valores brasileiro.

Posteriormente visando alinhar expectativas para se maximizar a possibilidade de se alcançar o objetivo pretendido, aplicou-se a técnica objetivo SMART no contexto do problema, conforme apresentado na próxima seção.

1.1.6 Objetivo SMART

- **Specific** (Específico): Identificar emoções expressas em Tweets.
- **Mensurable** (Mensurável): Utilizar métricas para estimar a eficiência do modelo nos resultados preditos.
- **Attainable** (Atingível): Solução que consiste em criar e treinar um modelo e fazer novas previsões através do teste com novos Tweets.
- **Relevant** (Relevante): Servir como indicador para corretoras de valores.
- **Time Based** (Temporal): Tempo de desenvolvimento das Sprints (6 meses aproximadamente).

1.1.7 Premissas e Restrições

Visando conhecer possíveis impactos que poderiam afetar o desenvolvimento do projeto foram levantados os potenciais restrições existentes, de forma a assegurar a conformidade nas fases de desenvolvimento planejadas. Para isso, mapeou-se as premissas e restrições e potenciais riscos que seguem apresentados na próxima tabela.

Premissas:

- Priorizar tecnologias *open source*;
- Priorizar tecnologias desenvolvidas em linguagem Python;

Restrições:

- Anonimizar usuários dos Tweets;

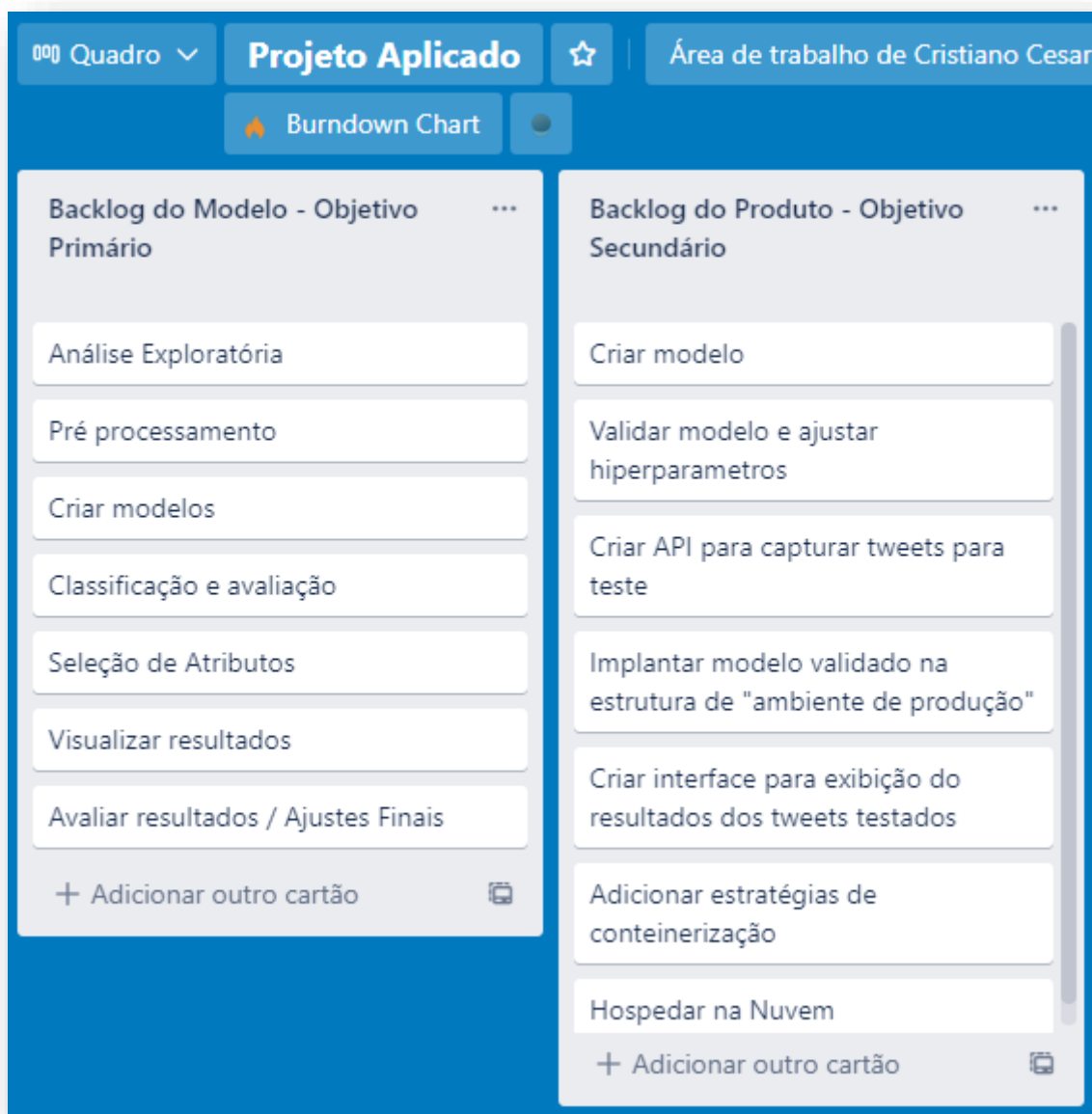
Tabela 6 - Matriz de Riscos

Matriz de Riscos			
Risco Identificado	Impacto Potencial	Ações preventivas	Ações corretivas
Integração de tecnologias	Alto	Pesquisar alternativas viáveis	Utilizar alternativas
Dependências na ordem de tarefas	Médio	Arquitetar solução	Utilizar metodologias Ágeis de Desenvolvimento

1.1.8 Backlog de Produto

Uma vez que foram estipulados objetivos primários e secundários para o projeto, segue a imagem extraída da plataforma Trello, que ilustra o backlog do que será desenvolvido.

Figura 3 - Backlog do produto



2. Área de Experimentação

2.1 Sprint 1

As capturas de telas a seguir demonstram a evolução do planejamento e execução do projeto na plataforma Trello, bem como apresenta evidências das execuções dos requisitos estabelecidos e suas respectivas soluções encontradas. Detalhes e desafios pertinentes a realização se encontram mais bem descritos na seção 2.1.2 Lições aprendidas.

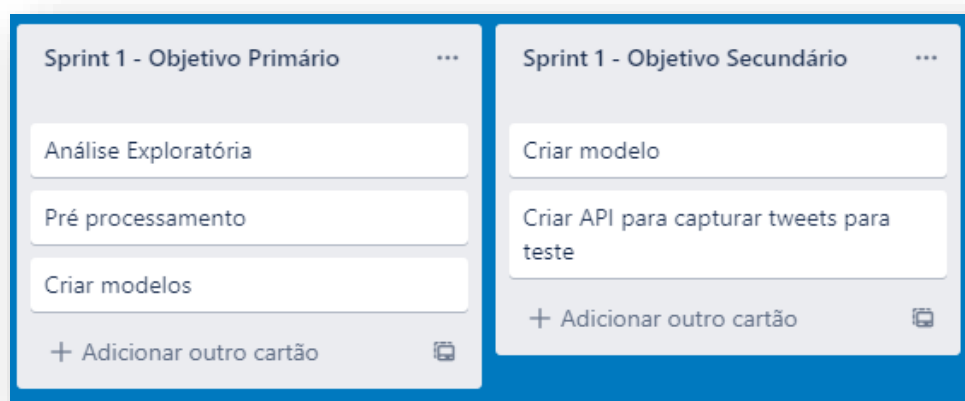
2.1.1 Solução

- **Evidência do planejamento:**

Figura 4 - Visão Geral dos Cards



Figura 5 - Visão específica dos cards da Sprint 1



- Evidência da execução de cada requisito:

Figura 6 - Evidência: Análise Exploratória

```
[ ] # Amostra do dataframe
df1.head()

# Dimensões (linhas x colunas) do dataframe
df1.shape

[ ] # Informações das variáveis do dataframe (nomes dos campos, quantidades nulas e não nulas, tipos de dados)
df1.info()

[ ] # Numero de instâncias classificadas de cada tipo
df1['sentiment'].value_counts()

[ ] # Estatísticas básicas das variáveis numéricas do dataframe
df1.describe(include="all")

[ ] #somatório de dados faltantes por coluna
df.isna().sum()
```

Figura 7 - Evidência: Pré-Processamento

```
▼ Pré-processamento dos dados

[ ] #duplicando nova coluna com o texto dos tweets que serão tratados
df1['new_text'] = df1['tweet_text'].copy()
df2['new_text'] = df2['text'].copy()

Remover Pontuação

[ ] df1['new_text'] = df1['new_text'].str.replace('[.,:;!]+', ' ', regex=True).copy()
df2['new_text'] = df2['new_text'].str.replace('[.,:;!]+', ' ', regex=True).copy()

Remover Caracteres especiais

[ ] df1['new_text'] = df1['new_text'].str.replace ('[/<>()|\\+\\-\\$%&#@\\'\\"]+', ' ', regex=True).copy()
df2['new_text'] = df2['new_text'].str.replace ('[/<>()|\\+\\-\\$%&#@\\'\\"]+', ' ', regex=True).copy()

Remover Números

[ ] df1['new_text'] = df1['new_text'].str.replace('[0-9]+', '', regex=True).copy()
df2['new_text'] = df2['new_text'].str.replace('[0-9]+', '', regex=True).copy()
```

Figura 8 - Evidência: Criação de Modelos

Modelos

Modelo 01

```
[ ] # Criando modelo
    clf = LinearSVC()
    # Treinamento do modelo
    clf.fit(X_train, y_train)
```

Modelo 02

```
# Criando modelo
mnb = MultinomialNB()
# Treinamento do modelo
mnb.fit(X_train, y_train)
```

Figura 9 - Evidência: Criação API Twitter

```
# Definir um arquivo de saída para armazenar os tweets coletados
data_hoje = datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
out = open(f"colleted_tweets_{data_hoje}.txt", "w")

# Implementar uma classe para conexão com o Twitter
class MyListener(StreamListener):

    def on_data(self, data):
        itemString = json.dumps(data)
        out.write(itemString + "\n")
        return True

    def on_error(self, status):
        print(status)

# Criando uma lista de palavras chave para buscar nos Tweets
keywords = ["BOVA11", "IVVB11", "PETR3", "VALE3", "BBDC4", "ABEV3", "ITUB4", "MGLU3", "CIEL3", "Ibovespa"]

# Implementar função MAIN
if __name__ == "__main__":
    l = MyListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)

    stream = Stream(auth, l)
    stream.filter(track=keywords)
```

- Evidência da solução:

Figura 10 – Evidência: Exemplo de resultados obtidos com a Análise Exploratória

```
# Dimensões (linhas x colunas) do dataframe
df1.shape

(785814, 5)

# Informações das variáveis do dataframe (nomes dos campos, quantidades nulas e não nulas, tipos de dados)
df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 785814 entries, 0 to 785813
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0    id          785814 non-null  int64
1    tweet_text  785814 non-null  object
2    tweet_date  785814 non-null  object
3    sentiment   785814 non-null  object
4    query_used  785814 non-null  object
dtypes: int64(1), object(4)
memory usage: 30.0+ MB

# Numero de instâncias classificadas de cada tipo
df1['sentiment'].value_counts()

Negativo    522707
Positivo    263107
Name: sentiment, dtype: int64
```

Figura 11 - Evidência: Exemplo de resultados obtidos com a execução da fase de Pré-Processamento

```
Remove Pontuação

[9] df1['new_text'] = df1['new_text'].str.replace('[.,:;!]+', ' ', regex=True).copy()
    df2['new_text'] = df2['new_text'].str.replace('[.,:;!]+', ' ', regex=True).copy()

Remove Caracteres especiais

[10] df1['new_text'] = df1['new_text'].str.replace ('[/<>()|\\+\\-\\$%&#@\\'\\"]+', ' ', regex=True).copy()
    df2['new_text'] = df2['new_text'].str.replace ('[/<>()|\\+\\-\\$%&#@\\'\\"]+', ' ', regex=True).copy()

Remove Números

[11] df1['new_text'] = df1['new_text'].str.replace('[0-9]+', '', regex=True).copy()
    df2['new_text'] = df2['new_text'].str.replace('[0-9]+', '', regex=True).copy()

df1.loc[15, ['tweet_text', 'new_text']]

tweet_text    @Calebe80781696 Vixe q treta novelesca! Que bo...
new_text      Calebe Vixe q treta novelesca  Que bom a amiz...
Name: 15, dtype: object
```



2.1.2 Lições aprendidas

Para a execução da Sprint 1 foram feitas algumas modificações, bem como aprendidas algumas lições, que merecem ser pontuadas nesta seção, sendo elas:

- Foi adicionado outro *dataset* base utilizado para o treinamento, pelas seguintes justificativas: o novo *dataset* utilizado denominado “*Portuguese Tweets for Sentiment Analysis*”, primeiramente possui uma quantidade bem superior de instâncias, o que tende a refletir em melhores resultados alcançados na etapa de treinamento dos modelos supervisionados, e segundo, por possuir apenas uma variável a ser predita, diferentemente do *dataset* “*Brazilian Stock Market Tweets with Emotions*”, que além de um número mais baixo de instâncias, possuía demasiados campos que correspondiam a diferentes sentimentos, o que, por sua vez, demandaria uma identificação de vários rótulos pelos modelos, todavia, os tweets deste *dataset* ainda serão utilizados na etapa de predição dos modelos.
- No processo de pré-processamento do texto, a aplicação de técnicas de *stemming* (redução de um termo ao seu radical), além onerosa a nível de tempo de processamento e custo computacional devido a quantidade de instancias do dataset utilizado, também diminuiu nos testes iniciais a acurácia alcançada pelos modelos, motivo pelo qual foi retirada do pré-processamento.
- A construção de modelos e da API de coleta de dados Twitter nas suas fases atuais foram baseadas, além do conteúdo apresentado no curso até o momento, em documentações oficiais e em outros tutoriais da internet.

Links dos notebooks:

Modelo:

https://colab.research.google.com/drive/1q4iSaridUN6R2n5lVn8QD7VH0rI_GiaQ?authuser=0#scrollTo=7AI7fAQtes-6

API Twitter para coleta de dados:

https://colab.research.google.com/drive/16Aif6rn0Mlz2c6TYn_mNrGhsdixb3iPE

2.2 Sprint 2

Em continuidade ao realizado na Sprint 1, as capturas de telas a seguir demonstram a evolução do planejamento e execução do projeto na plataforma Trello, bem como apresenta evidências das execuções dos requisitos estabelecidos e suas respectivas soluções encontradas. Detalhes e desafios pertinentes a realização se encontram mais bem descritos na seção 2.2.2 Lições aprendidas.

2.2.1 Solução

- Evidência do planejamento:

Figura 14 - Visão geral do estado dos Cards

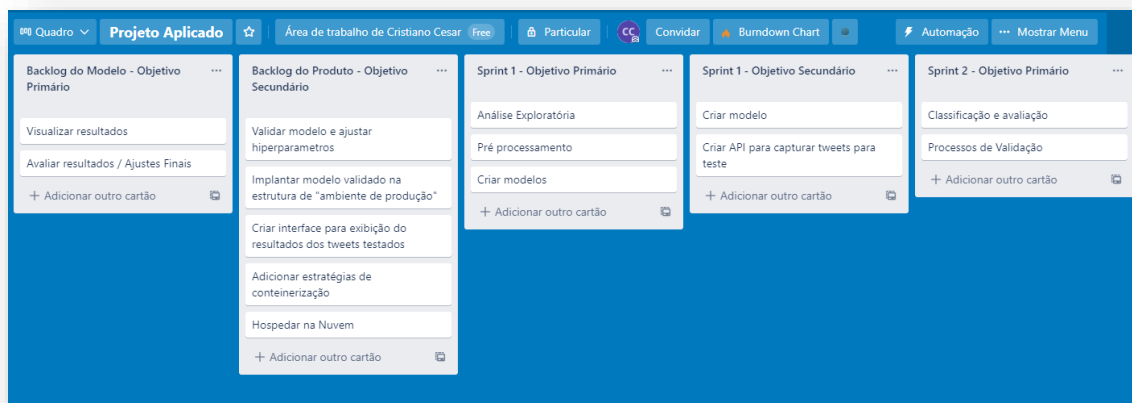
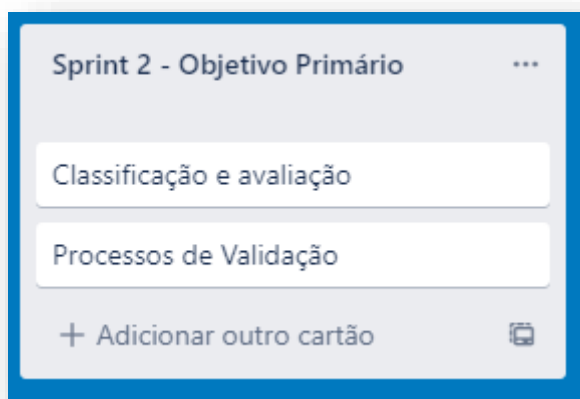


Figura 15 - Visão específica dos cards da Sprint 2



- Evidência da execução de cada requisito:

Figura 16 - Evidência: Classificação e Métricas da execução do modelo SVM

```
[17] # Realizando a predição
      predictions_clf = clf.predict(X_test)
      #predictions_clf1 = clf1.predict(X_test1)
      #predictions_clf2 = clf2.predict(X_test2)
      #predictions_clf3 = clf3.predict(X_test3)

#avaliando o modelo
print('Matriz de Confusão\n', metrics.confusion_matrix(y_test, predictions_clf))
print('\nAcurácia\n', metrics.accuracy_score(y_test, predictions_clf))
print('\nAcurácia Balanceada por classe\n', metrics.balanced_accuracy_score(y_test, predictions_clf))
print('\nPrecision\n', metrics.precision_score(y_test, predictions_clf))
print('\nRecall\n', metrics.recall_score(y_test, predictions_clf))
print('\nF1\n', metrics.f1_score(y_test, predictions_clf))
print('\nAUCROC\n', metrics.roc_auc_score(y_test, predictions_clf))
```

Figura 17 – Evidência: Métricas da execução do modelo Multinomial Naive Bayes

```
[22] # Realizando a predição
      predictions_mnb = mnb.predict(X_test)

#avaliando o modelo
print('Matriz de Confusão\n', metrics.confusion_matrix(y_test, predictions_mnb))
print('\nAcurácia\n', metrics.accuracy_score(y_test, predictions_mnb))
print('\nAcurácia Balanceada por classe\n', metrics.balanced_accuracy_score(y_test, predictions_mnb))
print('\nPrecision\n', metrics.precision_score(y_test, predictions_mnb))
print('\nRecall\n', metrics.recall_score(y_test, predictions_mnb))
print('\nF1\n', metrics.f1_score(y_test, predictions_mnb))
print('\nAUCROC\n', metrics.roc_auc_score(y_test, predictions_mnb))
```


Figura 18 - Evidência: Execução de validação por Treino e Teste

Treino-Teste

Dados e Divisão

```

# A entrada será a transformação de vetores com a normalização tf-idf
data = X_tfi
# A saída será os departamentos
labels = df1['sentiment']
# aplicando a funcao train_test_split para separar os conjuntos de treino e
# teste segundo uma porcentagem de separação definida.
# Separando 20% dos dados para teste
#X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.10, random_state = 42)
X_train1, X_test1, y_train1, y_test1 = train_test_split(data, labels, test_size=0.10, random_state = 42)
X_train2, X_test2, y_train2, y_test2 = train_test_split(data, labels, test_size=0.20, random_state = 42)
X_train3, X_test3, y_train3, y_test3 = train_test_split(data, labels, test_size=0.30, random_state = 42)

```

Figura 19 - Evidência: Execução de Validação por Validação Cruzada

Validação Cruzada

Dados

[24]

```

data = X_tfi
labels = df1['sentiment']

```

[25]

```

# aplicando o modelo de validação cruzada
# divide o dataset entre 5 diferentes grupos
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

```

Modelo

```

classifier_cv = LinearSVC()
scores_cv = cross_val_score(classifier_cv, data, labels, cv=kfold)
scores_cv

```

- Evidência da solução:

Figura 20 - Evidência: Resultado de diferentes métricas alcançadas pelo modelo SVM

```

Matriz de Confusão
[[47217  5069]
 [ 9371 16925]]

Acurácia
0.8162429054999872

Acurácia Balanceada por classe
0.7733432275569947

Precision
0.7695280531053924

Recall
0.6436340127776088

F1
0.7009732863946988

AUCROC
0.7733432275569948
  
```

Figura 21 - Evidência: Resultado de diferentes métricas alcançadas pelo modelo Multinomial Naive Bayes

```

Matriz de Confusão
[[51997   289]
 [21308  4988]]

Acurácia
0.7251660685653203

Acurácia Balanceada por classe
0.5920796760779835

Precision
0.9452340344892931

Recall
0.18968664435655613

F1
0.31596617362936685

AUCROC
0.5920796760779835
  
```

2.2.2 Lições aprendidas

Para a execução da Sprint 2 foram feitas algumas modificações, bem como aprendidas algumas lições, que merecem ser pontuadas nesta seção, sendo elas:

- A codificação das métricas para algoritmos de classificação e dos processos de validação foram aproveitadas do material apresentado no Bootcamp Analista de Machine Learning do IGTI, e ajustadas sendo feitas as adaptações pertinentes.
- Nos modelos de classificação foram testados os métodos de classificação SVM (*Support Vector Machine*), *Multinomial Naive Bayes* e o *Random Forest Classifier*, destes o que apresentou melhor desempenho foi o algoritmo SVM, com desempenho se pautando pela acurácia por volta de 80%. O modelo que utilizava árvores de decisão randômicas apesar de ter um desempenho similar ao modelo Naive Bayes apresentava uma latência muito alta no tempo de treinamento, o que faz do seu desempenho o pior entre os 3 classificadores testados até o momento.
- A tarefa no Card “Seleção de Atributos” foi renomeada para “Processos de Validação”, uma vez só existe um campo preditor no dataset “new_text”, não havendo assim que se falar em seleção de atributos, todavia a aplicação de diferentes modelos de validação é totalmente pertinente para a construção do modelo em questão e a tarefa que foi realizada na Sprint.
- Através da validação treino teste, foi observado que o tamanho do corte, em 10%, 20% e 30% não apresentou diferença significativa aos resultados atingidos pelos modelos.
- Até o momento a validação cruzada foi aplicada apenas ao modelo SVM, justamente por este apresentar resultados relativamente superiores na validação treino teste.
- Apesar de existir a tarefa de validação no Card de objetivos secundários, não foi realizada a sintonização de hiper parâmetros motivo pelo qual o card não foi deslocado do backlog.

Mesmo se tratando do mesmo modelo segue o link atualizado do notebook:

https://colab.research.google.com/drive/1q4iSaridUN6R2n5lvn8QD7VH0rl_GiaQ?usp=sharing

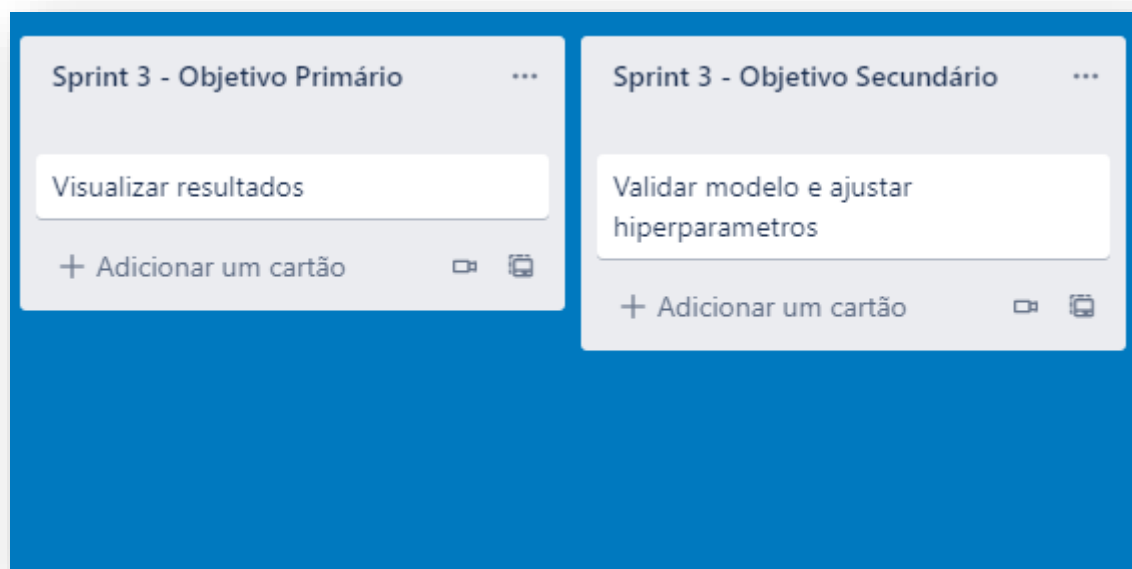
2.3 Sprint 3

Em continuidade ao realizado na Sprint 2, as capturas de telas a seguir demonstram a evolução do planejamento e execução do projeto na plataforma Trello, bem como apresenta evidências das execuções dos requisitos estabelecidos e suas respectivas soluções encontradas. Detalhes e desafios pertinentes a realização se encontram mais bem descritos na seção 2.3.2 Lições aprendidas.

2.3.1 Solução

- Evidência do planejamento:

Figura 22 - Visão Específica dos Cards da Sprint 3



- Evidência da execução de cada requisito:

Figura 23 - Evidência: Comparando diferentes algoritmos de classificação através da validação cruzada

```
[ ] # construindo os modelos de classificação
modelos = [LinearSVC(), MultinomialNB(), RandomForestClassifier(n_estimators= 10),KNeighborsClassifier(n_neighbors=3),
            SVC(kernel='sigmoid'), DecisionTreeClassifier(), LogisticRegression(), SGDClassifier()]

[ ] #utilizando a validação cruzada
mean=[]
std=[]
for model in modelos:
    result = cross_val_score(model, data, labels, cv=kfold, scoring='f1', n_jobs=-1)
    mean.append(result)
    std.append(result)

classificadores=['Linear SVM','Naives Bayes', 'Random Forest','KNN', 'SVM', 'Decision Tree', 'Logistic Regression', 'SGD']

plt.figure(figsize=(15, 10))
for i in range(len(mean)):
    sns.distplot(mean[i], hist=False, kde_kws={"shade": True})

plt.title("Distribuição de cada um dos classificadores", fontsize=15)
plt.legend(classificadores)
plt.xlabel("F1", labelpad=20)
plt.yticks([])

plt.show()
```

Figura 24 - Evidência: Ajustando hiper parâmetros de um modelo através do Grid Search

```
Grid Search

#definindo hiperparâmetros
hiperparam = {'kernel':('sigmoid', 'rbf'), 'C':[0.01, 1, 10]}

#definindo o tipo de validacao cruzada e o numero de folds
cv_strat = StratifiedKFold(n_splits = 10)

#instanciando meu classificador
classifier = SVC()

#definindo a estrategia de score a partir da metrica f1
f1 = make_scorer(f1_score)

#instanciando e modelando o grid search com os hiperparametros e a validação definidas.
grid_cv = GridSearchCV(classifier, hiperparam, cv = cv_strat, scoring = f1)
grid_cv.fit(data, labels)
```

Figura 25 - Evidência: Ajustando hiperparâmetros de um modelo através do Random Search

```

▼ Random Search

[ ] #definindo o tipo de validacao cruzada e o numero de folds
    cv_strat = StratifiedKFold(n_splits = 10)

    #definindo a estrategia de score a partir da metrica f1
    f1 = make_scorer(f1_score)

    #definindo hiperparâmetros
    distributions = dict(kernel = ['sigmoid', 'rbf'],
                        C = uniform(loc=0, scale=10))

    #instanciando meu classificador
    classifier = SVC()

    #instanciando e modelando o random search com os hiperparametros e a validação definidas.
    random_cv = RandomizedSearchCV(classifier, distributions, cv = cv_strat, scoring = f1, random_state = 42, n_iter = 10)
    random_cv.fit(data, labels)
    
```

- Evidência da solução:

Figura 26 - Comparativo da média da métrica F1 Score em aplicada em diferentes algoritmos de classificação

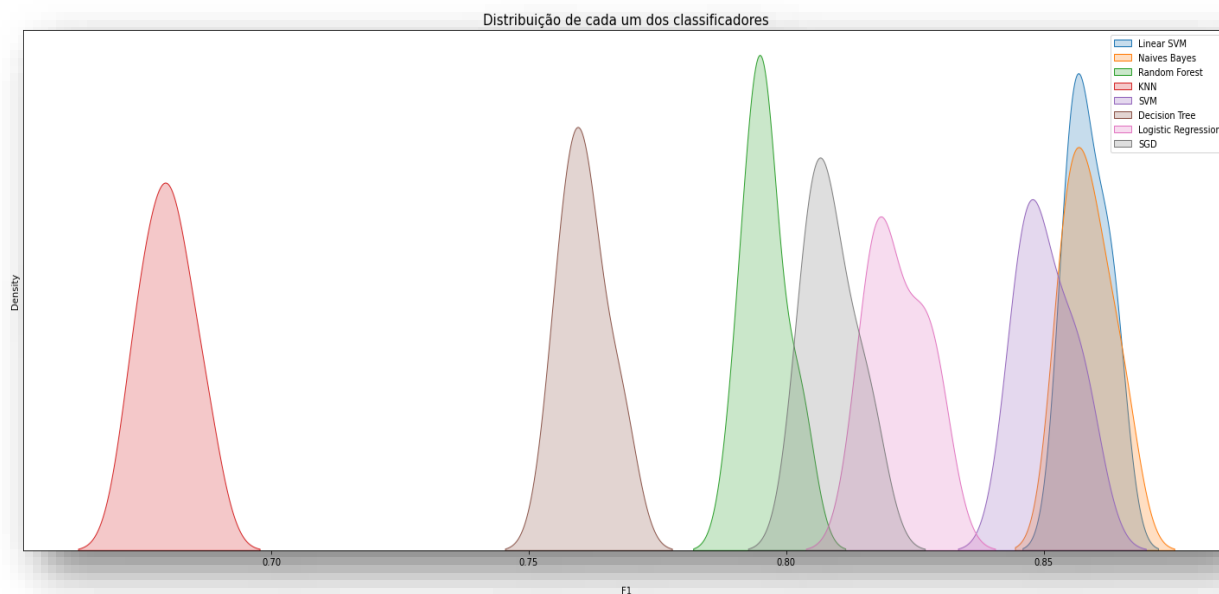


Figura 27 - Resultado do melhor modelo encontrado pelo Grid Search utilizando o classificador SVM.

```
[ ] #vamos olhar para os melhores resultados encontrados pelo Grid Search
print('Melhor resultado f1:', grid_cv.best_score_)
print('\n\nMelhor configuração de hiperparâmetros:', grid_cv.best_params_)

print( '\n\nConfigurações de todos os hiperparâmetros do melhor estimado encontrado pelo GridSearch: \n', grid_cv.best_estimator_)

Melhor resultado f1: 0.8811811907521531

Melhor configuração de hiperparâmetros: {'C': 1, 'kernel': 'sigmoid'}

Configurações de todos os hiperparâmetros do melhor estimado encontrado pelo GridSearch:
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figura 28 - Resultado do melhor modelo encontrado pelo Random Search utilizando o classificador SVM.

```
[ ] #vamos olhar para os melhores resultados encontrados pelo Random Search
print('Melhor resultado f1:', random_cv.best_score_)
print('\n\nMelhor configuração de hiperparâmetros:', random_cv.best_params_)
print( '\n\nConfigurações de todos os hiperparâmetros do melhor estimado encontrado pelo Random Search: \n', random_cv.best_estimator_)

Melhor resultado f1: 0.8988574129384126

Melhor configuração de hiperparâmetros: {'C': 3.745401188473625, 'kernel': 'sigmoid'}

Configurações de todos os hiperparâmetros do melhor estimado encontrado pelo Random Search:
SVC(C=3.745401188473625, break_ties=False, cache_size=200, class_weight=None,
    coef0=0.0, decision_function_shape='ovr', degree=3, gamma='scale',
    kernel='sigmoid', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

2.3.2 Lições aprendidas

Para a execução da Sprint 3 foram feitas algumas modificações, bem como aprendidas algumas lições, que merecem ser pontuadas nesta seção, sendo elas:

- Na etapa de pré-processamento, como novo teste, houve nova aplicação de técnicas de *stemming*, mas diferentemente da Sprint 1 em que foi utilizada uma função *lambda* nativa do Python, na atual Sprint optou-se por construir uma função própria para esta finalidade, otimizando o tempo de aplicação gasto no processo de aplicação desta em todas as amostras do *dataset*. Todavia, a aplicação desta técnica não impactou em melhores resultados pelos classificadores.
- Visando minimizar distorções nos resultados alcançados, para resolver o problema de desbalanceamento de dados que afetava a distribuição das classes no *dataset* em uma proporção de 70/30, foram aplicadas técnicas de *oversampling* de forma a igualar o número de amostras por classe. Balanceamento que melhorou os resultados alcançados por métricas como F1 Score e AucRoc.
- Foi feita também uma comparação de diferentes técnicas de classificação visando encontrar os algoritmos com as melhores performances, em que se comparou a média da métrica F1 Score alcançada pela aplicação de validação cruzada nas diferentes técnicas em sua configuração padrão de parâmetros. Desta comparação se constatou que as técnicas com melhores performance foram a SVM, Linear SVM e Naive Bayes.
- A partir da comparação, se realizou o *model tuning* pela aplicação de técnicas como *Grid Search* e *Random Search*, visando encontrar a melhor combinação de hiper parâmetros.
- Finalmente, foi realizada a exportação um modelo e suas configurações considerado ótimo pelas técnicas de *model tuning* para um posterior *deploy* em uma simulação operação de um ambiente de produção.

Mesmo se tratando do mesmo modelo segue o link atualizado do notebook:

https://colab.research.google.com/drive/1q4iSaridUN6R2n5lvn8QD7VHOrl_GiaQ?usp=sharing

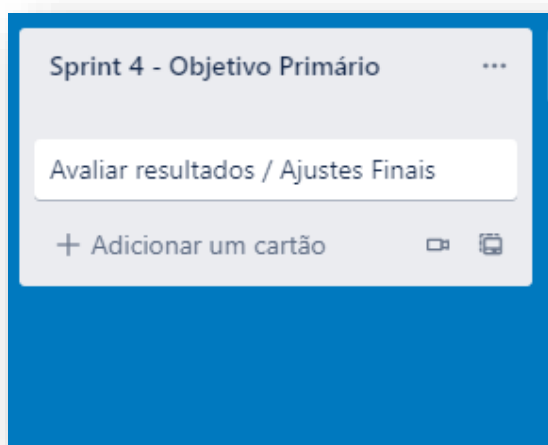
2.4 Sprint 4

Em continuidade ao realizado na Sprint 3, as capturas de telas a seguir demonstram a evolução do planejamento e execução do projeto na plataforma Trello, bem como apresenta evidências das execuções dos requisitos estabelecidos e suas respectivas soluções encontradas. Detalhes e desafios pertinentes a realização se encontram mais bem descritos na seção 2.4.2 Lições aprendidas.

2.4.1 Solução

- Evidência do planejamento:

Figura 29 - Visão geral dos cards da Sprint 4



- Evidência da execução de cada requisito:

Figura 30 - Evidência: Definição de variáveis flags para auxiliar na execução do modelo

```

LIB_LOAD = True #habilita/desabilita a carga das bibliotecas
DATA_LOAD = True #habilita/desabilita a carga do dataset
DEPLOY = False #habilita/desabilita a exportação do resultado do modelo
LEAVE_ONE_OUT_TRAIN = False #habilita/desabilita o treino pela técnica leave one out
GRID_SEARCH_TRAIN = False #habilita/desabilita a sintonização pela técnica Grid Search
RANDOM_SEARCH_TRAIN = False #habilita/desabilita a sintonização pela técnica Random Search
NUMBER_OF_SAMPLES = 10000 #Define o número de amostras a serem carregadas do dataset
    
```

Figura 31 - Evidência: Melhores parâmetros pela sintonização de hiper parâmetros

```

[154] #Configurações de todos os hiperparâmetros do melhor estimado encontrado pelo Random Search:
      modelo = SVC(C=3.745401188473625, break_ties=False, cache_size=200, class_weight=None,
                  coef0=0.0, decision_function_shape='ovr', degree=3, gamma='scale',
                  kernel='sigmoid', max_iter=-1, probability=False, random_state=None,
                  shrinking=True, tol=0.001, verbose=False)
    
```

Figura 32 - Evidência: Exportando o melhor modelo encontrado

```

# Aplica o vetorizador nos dados de texto e retorna uma matriz esparsa ( contendo vários zeros):
if _DEPLOY:
    vec_file = '/content/gdrive/My Drive/Colab Notebooks/Projeto Final IGTI/model/vectorizer.pickle'
    pickle.dump(data, open(vec_file, 'wb'))
else:
    vec_file = '/content/gdrive/My Drive/Colab Notebooks/Projeto Final IGTI/model/vectorizer.pickle'
    pickle.dump(data, open(vec_file, 'wb'))
    vec_file = '/content/gdrive/My Drive/Colab Notebooks/Projeto Final IGTI/production/model/vectorizer.pickle'
    pickle.dump(data, open(vec_file, 'wb'))



[160] #Salva o modelo
if _DEPLOY:
    with open('/content/gdrive/My Drive/Colab Notebooks/Projeto Final IGTI/model/modelo.pkl', 'wb') as model:
        pickle.dump(modelo, model)
else:
    with open('/content/gdrive/My Drive/Colab Notebooks/Projeto Final IGTI/model/modelo.pkl', 'wb') as model:
        pickle.dump(modelo, model)
    with open('/content/gdrive/My Drive/Colab Notebooks/Projeto Final IGTI/production/app/model/modelo.pkl', 'wb') as model:
        pickle.dump(modelo, model)
    
```

- Evidência da solução:

Figura 33 - Evidência: Execução do modelo encontrado pela sintonização de hiper parâmetros

```
✓ [155] modelo.fit(data, labels)
14s
SVC(C=3.745401188473625, break_ties=False, cache_size=200, class_weight=None,
coef0=0.0, decision_function_shape='ovr', degree=3, gamma='scale',
kernel='sigmoid', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)
```

Figura 34 - Evidência: Arquivo gerados pela exportação do modelo

Nome ↓	Proprietário	Última modificação	Tamanho do arquivo
 vectorizer.pickle	eu	16:59 eu	2,1 MB
 modelo.pkl	eu	16:59 eu	1 MB

2.4.2 Lições aprendidas

Para a execução da Sprint 4 foram feitas algumas modificações, bem como aprendidas algumas lições, que merecem ser pontuadas nesta seção, sendo elas:

- Como forma de ajustes finais, foram definidas algumas variáveis *flags* para controle e execução de trechos específicos da codificação;
- Em continuidade a definição da exportação definida na Sprint anterior, nesta foi de fato realizada a exportação conforme evidências. Em que, uma vez encontrada uma sintonização considerada ótima de hiper parâmetros se realizou a exportação do modelo para uma posterior simulação de produção e avaliação do desempenho com dados desconhecidos pelo modelo.

Segue o link atualizado do notebook:

https://colab.research.google.com/drive/1q4iSaridUN6R2n5lvn8QD7VH0rl_GiaQ?usp=sharing

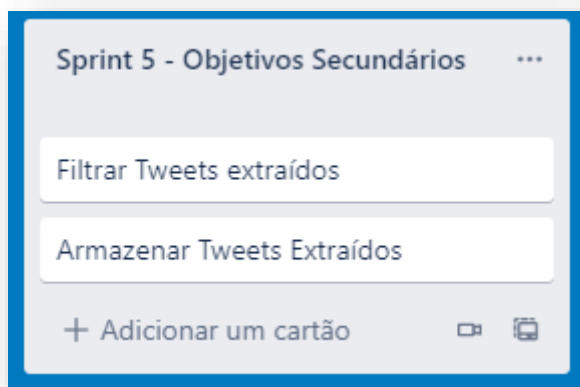
2.5 Sprint 5

Em continuidade ao realizado na Sprint 4, as capturas de telas a seguir demonstram a evolução do planejamento e execução do projeto na plataforma Trello, bem como apresenta evidências das execuções dos requisitos estabelecidos e suas respectivas soluções encontradas. Detalhes e desafios pertinentes a realização se encontram mais bem descritos na seção 2.5.2 Lições aprendidas.

2.5.1 Solução

- Evidência do planejamento:

Figura 35 - Visão geral dos cards da Sprint 5



- Evidência da execução de cada requisito:

Figura 36 - Evidência: Filtrando campos dos Tweets e Salvando em um DataFrame

```
import json
import pandas as pd

# Abrir o arquivo de Tweets e ler as linhas
with open('collected_tweets_2021-09-12-15-16-18.txt', 'r') as file:
    tweets = file.readlines()

parsed_tweets = [json.loads(json.loads(i)) for i in tweets]

def tweet_to_df(tweet):
    try:
        df_tratado = pd.DataFrame(tweet).reset_index(drop=True).iloc[:1]
        df_final = df_tratado[['id', 'text', 'created_at']]
        df_final = df_final.head(0)
        data = {'id': tweet['id'], 'text': tweet['text'], 'created_at': tweet['created_at']}
        df_final = df_final.append(data, ignore_index = True)

    except:
        return None

    return df_final

parseados = [tweet_to_df(tweet) for tweet in parsed_tweets]
parseados = [i for i in parseados if i is not None]
tratado = pd.concat(parseados, ignore_index=True)
```

Figura 37 - Evidência: Conectando e exportando dados para o MySQL

```
[ ] pip install mysql-connector-python
import mysql
import pyodbc
import sqlalchemy
import os
from sqlalchemy import create_engine


database_username = 'root'
database_password = '*****'
database_ip = 'localhost'
database_name = 'tweets_sentiment'

engine = sqlalchemy.create_engine('mysql+mysqlconnector://{0}:{1}@{2}/{3}'.format(database_username,
database_password,
database_ip,
database_name))

tratado.to_sql(name = 'twittes', con = engine, index = False, if_exists = 'append')
```

- Evidência da solução:

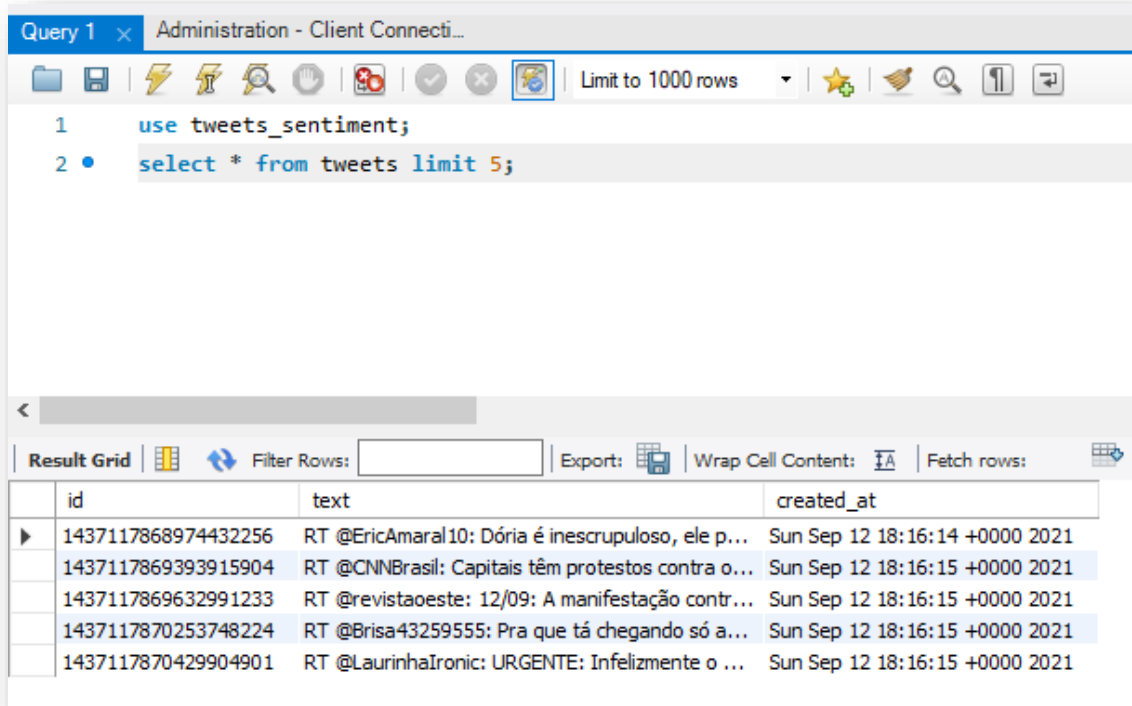
Figura 38 - Evidência: Tweets filtrados salvos na estrutura de um dataset



tratado.head()

	id	text	created_at
0	1437117868974432256	RT @EricAmaral10: Dória é inescrupuloso, ele p...	Sun Sep 12 18:16:14 +0000 2021
1	1437117869393915904	RT @CNNBrasil: Capitais têm protestos contra o...	Sun Sep 12 18:16:15 +0000 2021
2	1437117869632991233	RT @revistaoeste: 12/09: A manifestação contra...	Sun Sep 12 18:16:15 +0000 2021
3	1437117870253748224	RT @Brisa43259555: Pra que tá chegando só agora	Sun Sep 12 18:16:15 +0000 2021
4	1437117870429904901	RT @LaurinhaIronic: URGENTE: Infelizmente o MB...	Sun Sep 12 18:16:15 +0000 2021

Figura 39 - Evidência: Tweets armazenados em um banco de dados relacional



Query 1 Administration - Client Connecti...

```

1 use tweets_sentiment;
2 • select * from tweets limit 5;

```

Limit to 1000 rows

Result Grid

	id	text	created_at
▶	1437117868974432256	RT @EricAmaral10: Dória é inescrupuloso, ele p...	Sun Sep 12 18:16:14 +0000 2021
	1437117869393915904	RT @CNNBrasil: Capitais têm protestos contra o...	Sun Sep 12 18:16:15 +0000 2021
	1437117869632991233	RT @revistaoeste: 12/09: A manifestação contr...	Sun Sep 12 18:16:15 +0000 2021
	1437117870253748224	RT @Brisa43259555: Pra que tá chegando só a...	Sun Sep 12 18:16:15 +0000 2021
	1437117870429904901	RT @LaurinhaIronic: URGENTE: Infelizmente o ...	Sun Sep 12 18:16:15 +0000 2021

2.5.2 Lições aprendidas

Para a execução da Sprint 5 foram feitas algumas modificações, bem como aprendidas algumas lições, que merecem ser pontuadas nesta seção, sendo elas:

- Os Objetivos Secundários: Filtrar Tweets Extraídos e Armazenar Tweets Extraídos não listados no Backlog inicial, foram adicionados e realizados, uma vez que são requisitos necessários para continuação do projeto.
- Os campos filtrados do arquivo JSON gerado pela API do Twitter são equivalentes aos do *dataset* que foram utilizados para o treinamento inicial e criação do modelo.
- Como os dados filtrados foram armazenados em memória no formato de um *dataset* do Pandas, sua persistência se fez em uma base de dados também relacional no caso o MySQL, todavia não há restrições para armazenar os dados no formato JSON em um SGBD NOSql.

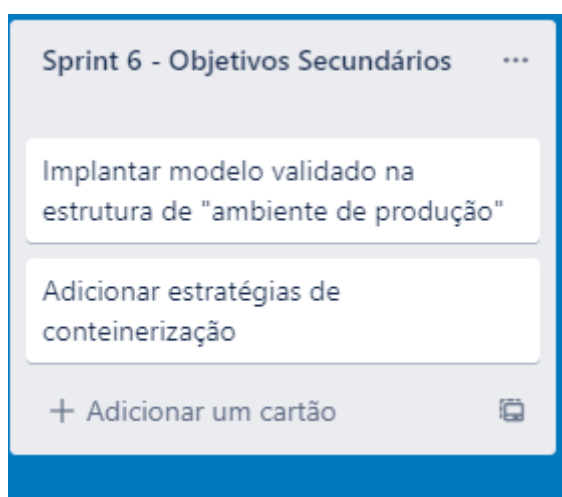
2.6 Sprint 6

2.6.1 Solução

Em continuidade ao realizado na Sprint 5, as capturas de telas a seguir demonstram a evolução do planejamento e execução do projeto na plataforma Trello, bem como apresenta evidências das execuções dos requisitos estabelecidos e suas respectivas soluções encontradas. Detalhes e desafios pertinentes a realização se encontram mais bem descritos na seção 2.6.2 Lições aprendidas.

- Evidência do planejamento:

Figura 40 - Visão geral do card da Sprint 6



- Evidência da execução de cada requisito:

Figura 41 - Evidência: trecho do arquivo com funções para processar e prever novos tweets em produção

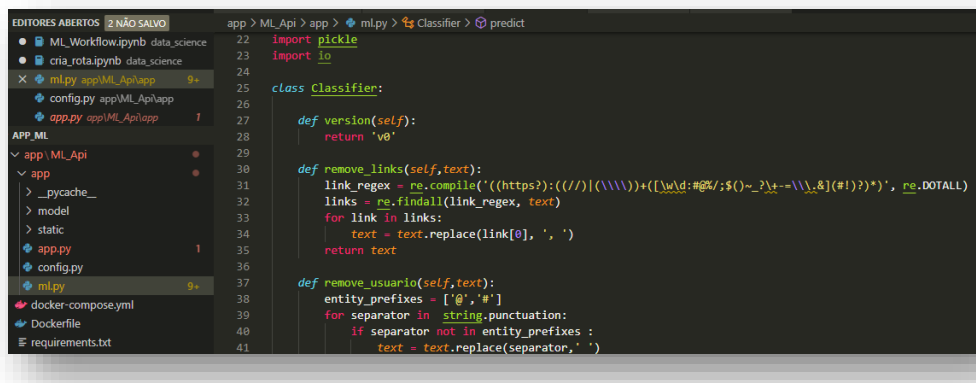


Figura 42 - Evidência: Criação de Interface através do Flask

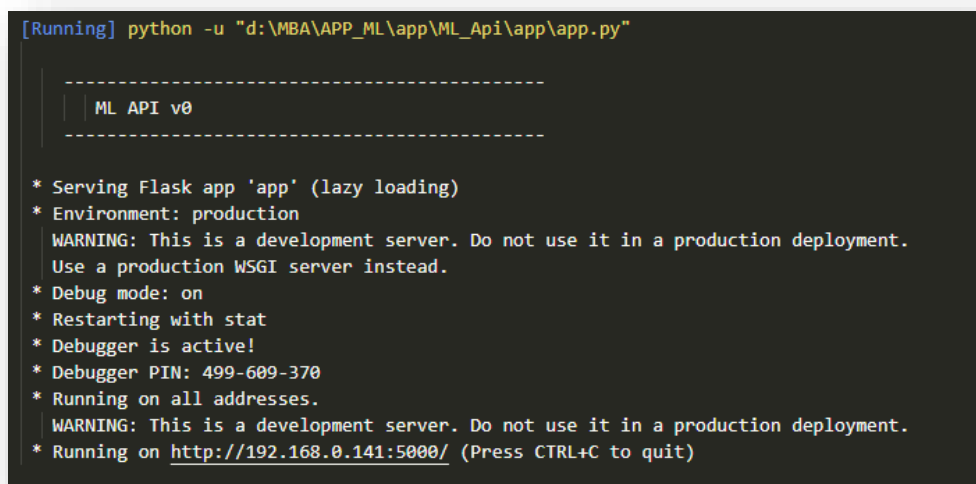
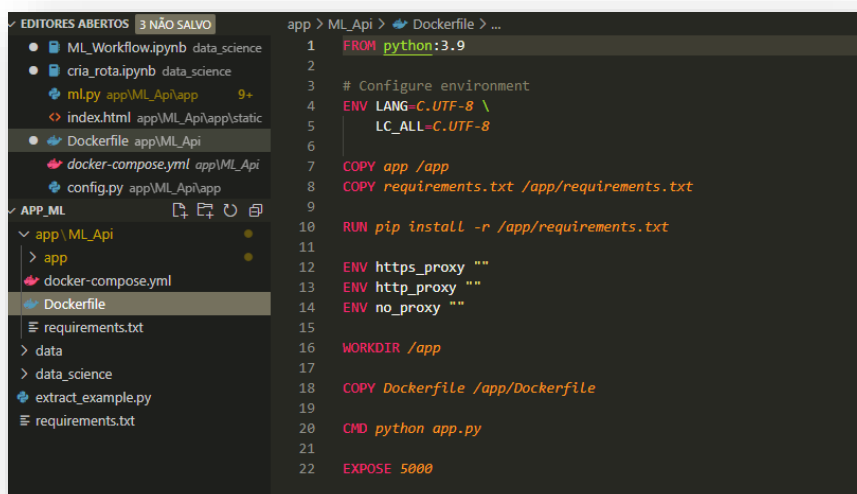


Figura 43 – Evidência: Algumas configurações para criação de uma imagem Docker da solução



- Evidência da solução:

Figura 44 - Evidência: Teste com Tweet predito como negativo

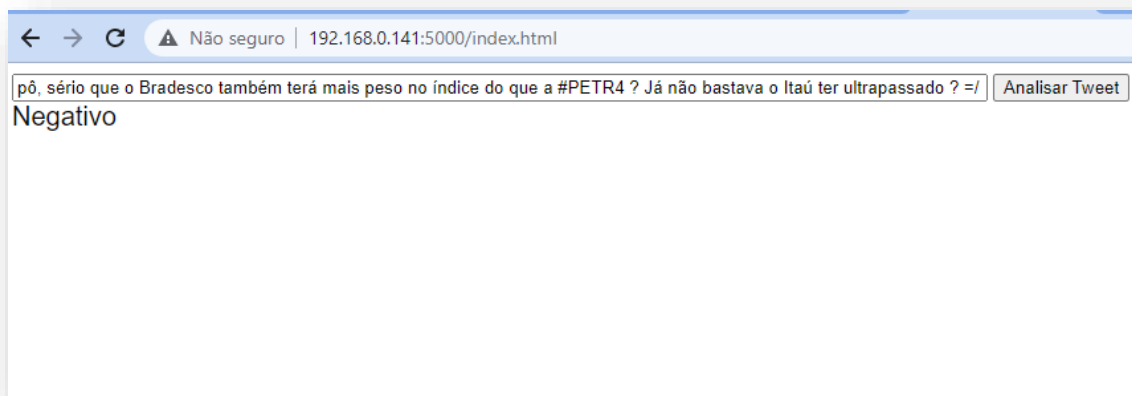
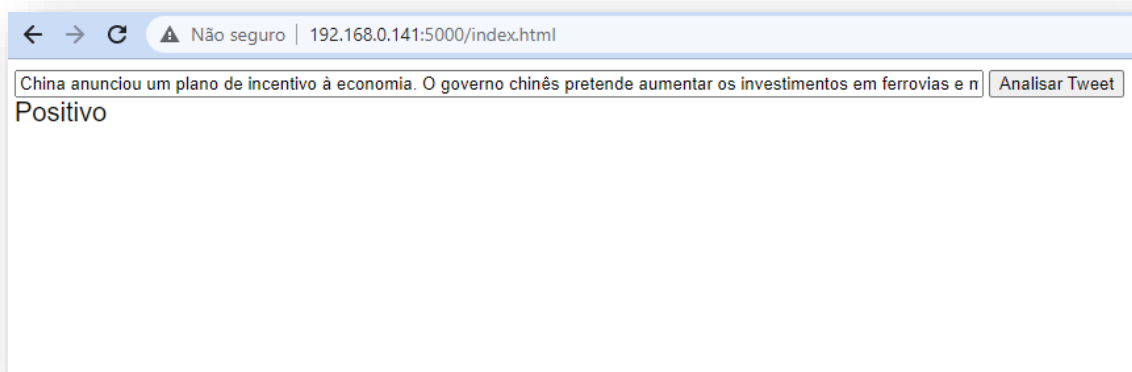


Figura 45 - Evidência: - Teste com Tweet predito como positivo



2.6.2 Lições aprendidas

Para a execução da Sprint 6 foram feitas algumas modificações, bem como aprendidas algumas lições, que merecem ser pontuadas nesta seção, sendo elas:

- As atividades de pré-processamento utilizadas de forma sequencial na criação do modelo dentro do jupyter notebook tiveram sua estrutura adequadas e foram dispostas em forma de funções em um script python para o seu uso em um ambiente de produção, juntamente com os modelos exportados, treinados pela configuração ótima do algoritmo SVC encontrada através do Grid Search.
- Os arquivos que compõem o modelo foram estruturados para que os serviços necessários operem dentro de uma imagem Docker, logo caso se queria subir a solução, essa pode rodar dentro de uma máquina virtual Linux, disponibilizada por um serviço de Cloud Computing.
- O serviço Flask responsável pela interface de operações e resultados já está funcionando para testes manuais, todavia, ainda se faz necessário fazer com que este plote os resultados de tweets capturados de uma forma gráfica que agregue informações estatísticas e relevantes ao usuário final.

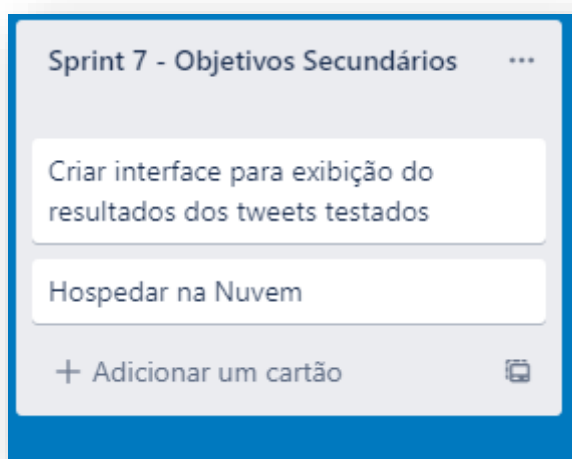
2.7 Sprint 7

2.7.1 Solução

Em continuidade ao realizado na Sprint 6, as capturas de telas a seguir demonstram a evolução do planejamento e execução do projeto na plataforma Trello, bem como apresenta evidências das execuções dos requisitos estabelecidos e suas respectivas soluções encontradas. Detalhes e desafios pertinentes a realização se encontram mais bem descritos na seção 2.7.2 Lições aprendidas.

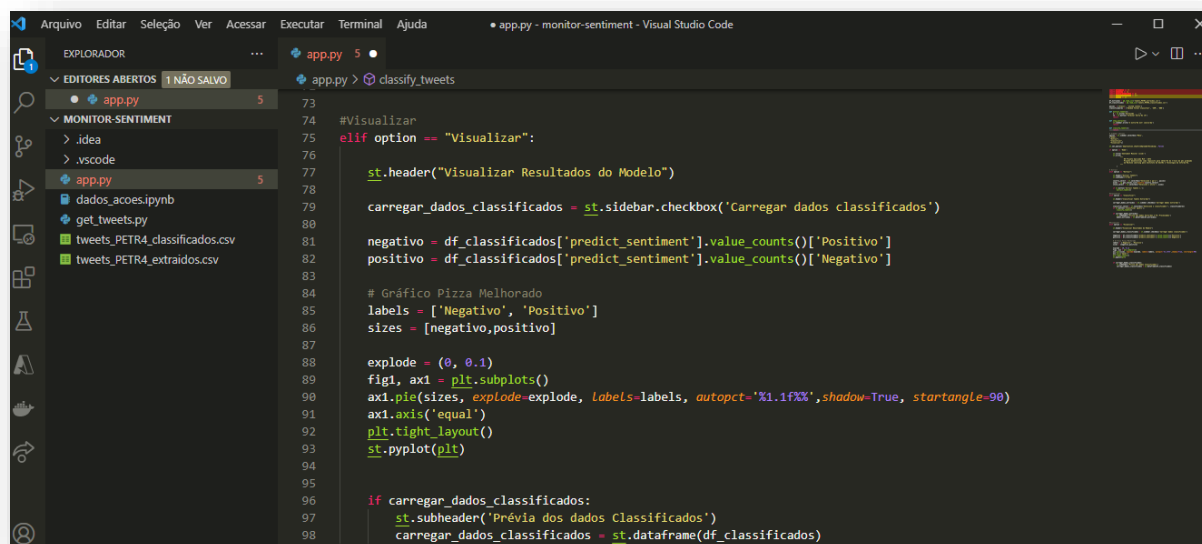
Evidência do planejamento:

Figura 46 - Visão geral do card da Sprint 7



- Evidência da execução de cada requisito:

Figura 47 - Evidência - Trecho do Código da Interface pelo Streamlit

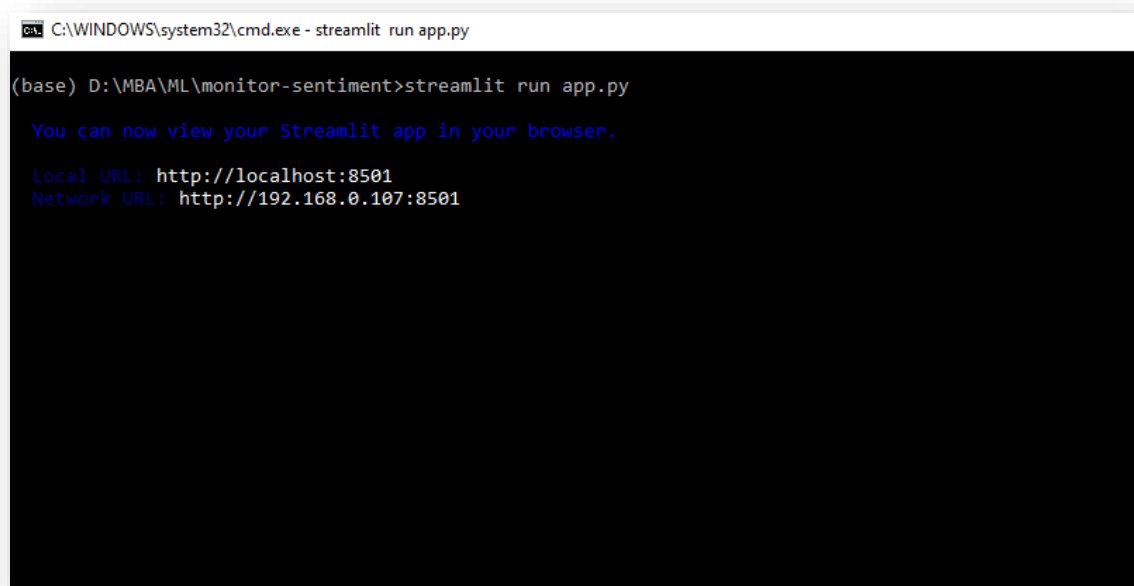


```

73
74 #Visualizar
75 elif option == "Visualizar":
76
77     st.header("Visualizar Resultados do Modelo")
78
79     carregar_dados_classificados = st.sidebar.checkbox("Carregar dados classificados")
80
81     negativo = df_classificados['predict_sentiment'].value_counts()['Positivo']
82     positivo = df_classificados['predict_sentiment'].value_counts()['Negativo']
83
84     # Gráfico Pizza Melhorado
85     labels = ['Negativo', 'Positivo']
86     sizes = [negativo, positivo]
87
88     explode = (0, 0.1)
89     fig1, ax1 = plt.subplots()
90     ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90)
91     ax1.axis('equal')
92     plt.tight_layout()
93     st.pyplot(plt)
94
95
96 if carregar_dados_classificados:
97     st.subheader("Prévia dos dados Classificados")
98     carregar_dados_classificados = st.dataframe(df_classificados)
99

```

Figura 48 – Evidência: Execução do Streamlit



```

C:\WINDOWS\system32\cmd.exe - streamlit run app.py

(base) D:\MBA\ML\monitor-sentiment>streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.107:8501

```

Figura 49 - Evidência: Teste de hospedagem do App no serviço Heroku

```
d:\MBA\ML\monitor-sentiment>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Procfile
    app.py
    requirements.txt
    setup.sh
    tweets_PETR4_classificados.csv
    tweets_PETR4_extraidos.csv

d:\MBA\ML\monitor-sentiment>git add .

d:\MBA\ML\monitor-sentiment>git commit -m "first commit"
[master (root-commit) b2feabd] first commit
6 files changed, 235 insertions(+)
create mode 100644 Procfile
create mode 100644 app.py
create mode 100644 requirements.txt
create mode 100644 setup.sh
create mode 100644 tweets_PETR4_classificados.csv
create mode 100644 tweets_PETR4_extraidos.csv

d:\MBA\ML\monitor-sentiment>git push heroku master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
```

- Evidência da solução:

Figura 49 - Evidência: Tela Inicial da aplicação em Streamlit

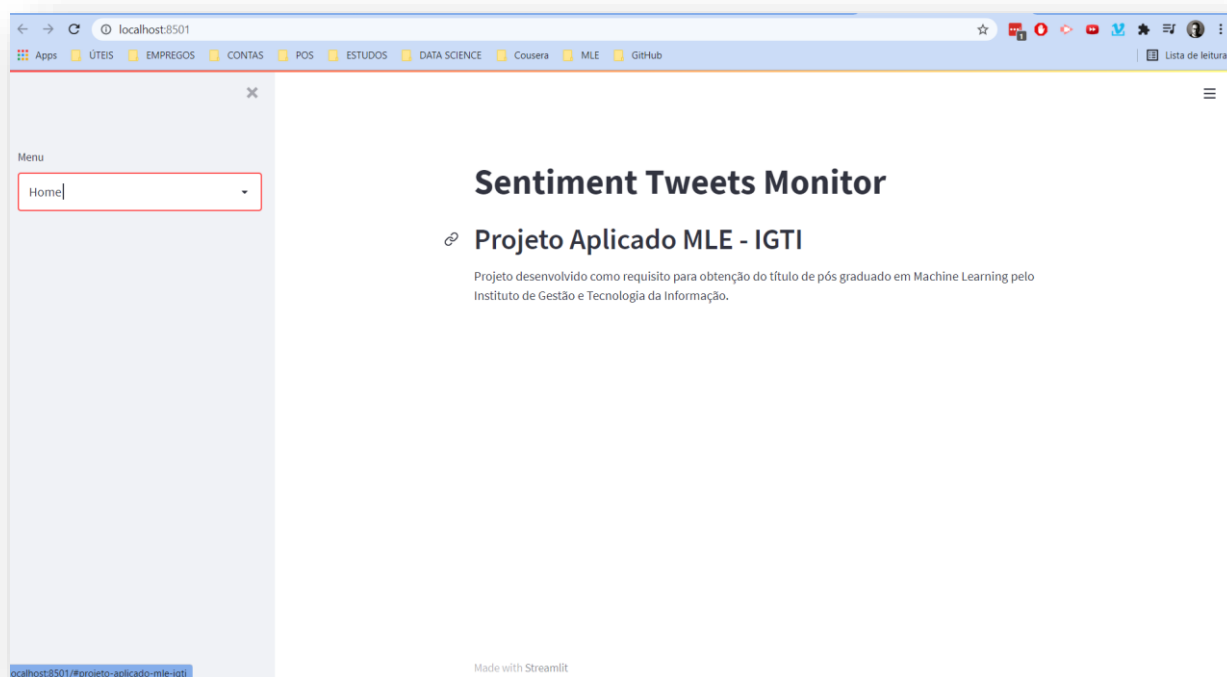


Figura 50 - Evidência: Tela de Extração de Tweets da aplicação em Streamlit

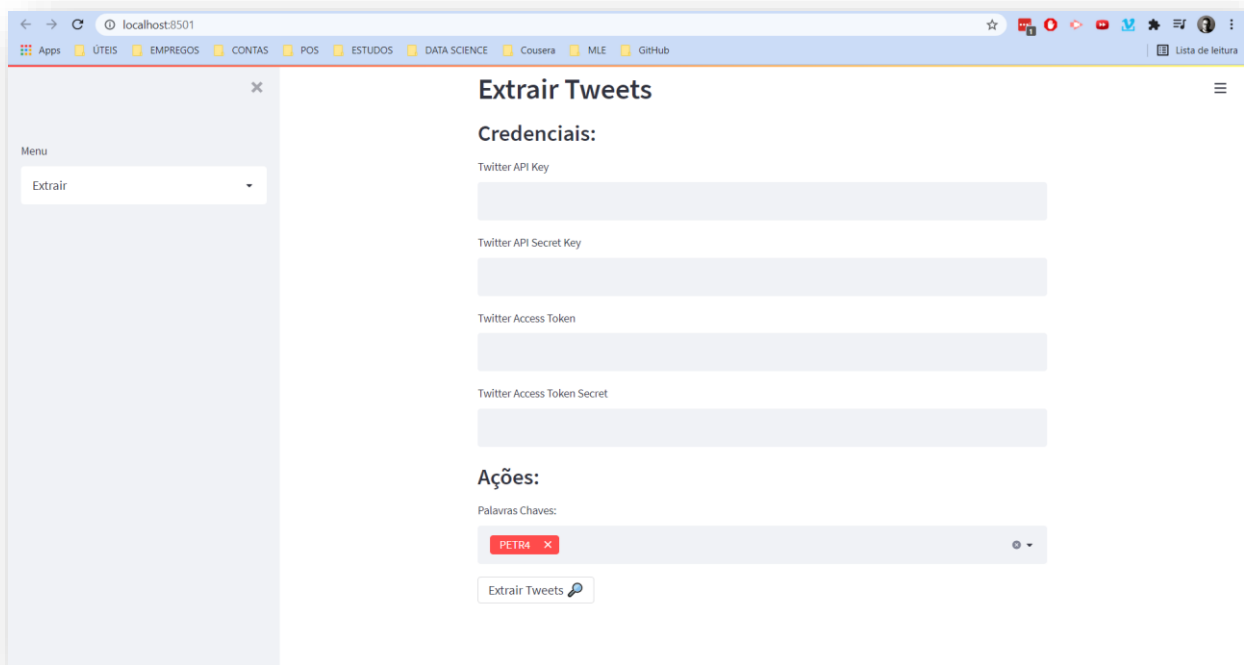


Figura 51 - Evidência: Tela de Classificação da aplicação em Streamlit

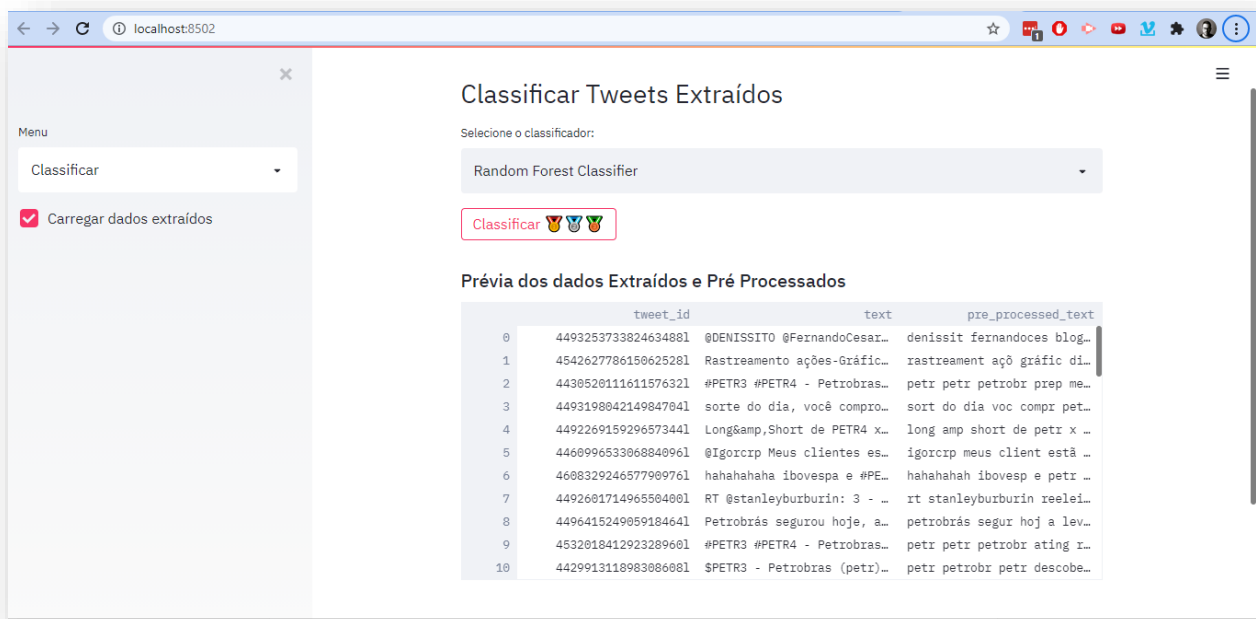


Figura 52 - Evidência: Tela de Visualização de Resultados da aplicação em Streamlit - parte 1

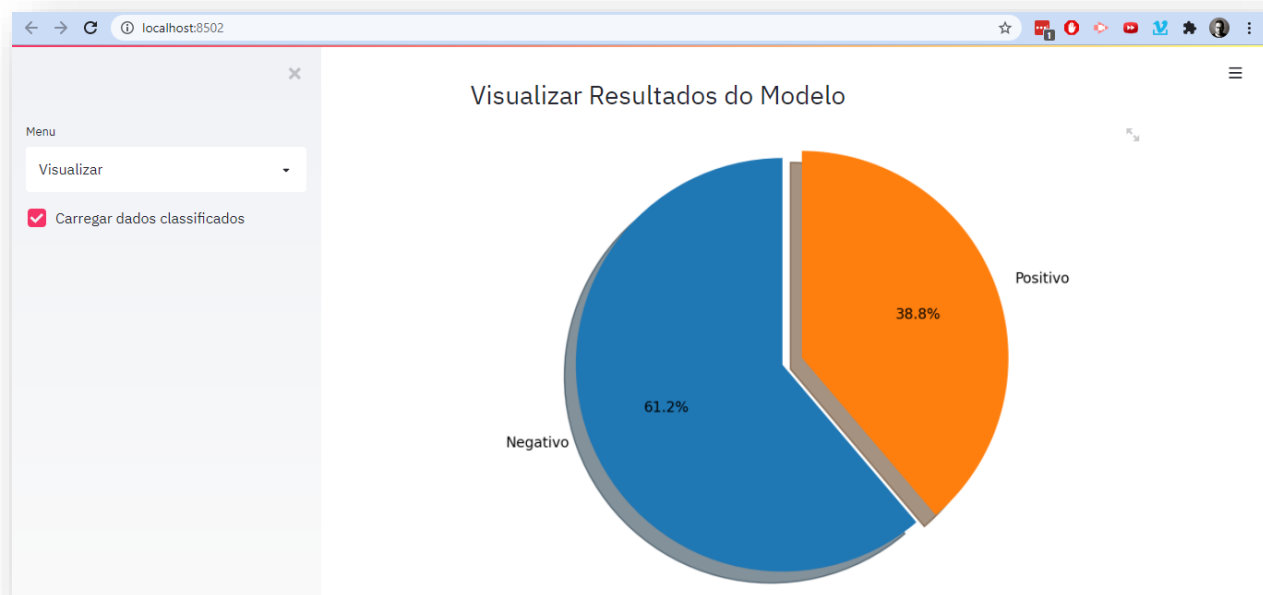


Figura 53 - Evidência: Tela de Visualização de Resultados da aplicação em Streamlit - parte 2



Prévia dos dados Classificados

tweet_id	text	pre_processed_text	predict_sentiment
0 33824634881	@DENISSITO @FernandoCe...	denissit fernandoces b...	Negativo
1 36150625281	Rastreamento ações-Grá...	rastreament açõ gráfico...	Positivo
2 11611576321	@PETR3 @PETR4 - Petrob...	petr petr petrobr prep...	Positivo
3 42149847041	sorte do dia, você com...	sort do dia voc compr ...	Positivo
4 59296573441	Long&Short de PETR...	long amp short de petr...	Negativo
5 33068840961	@Igorcixp Meus clientes...	igorcixp meus client es...	Negativo
6 46577909761	hahahahaha ibovespa e ...	hahahahah ibovesp e pe...	Negativo
7 14965504001	RT @stanleyburburin: 3...	rt stanleyburburin ree...	Positivo
8 49059184641	Petrobrás segurou hoje...	petrobrás segur hoj a ...	Positivo
9 12923289601	@PETR3 @PETR4 - Petrob...	petr petr petrobr atin...	Positivo
10 18983086081	\$PETR3 - Petrobras (pe...	petr petrobr petr desc...	Positivo

2.7.2. Lições aprendidas

Para a execução da Sprint 7 foram feitas algumas modificações, bem como aprendidas algumas lições, que merecem ser pontuadas nesta seção, sendo elas:

- Em novos testes com dados desconhecidos pelo modelo primário, observou-se que o desempenho do algoritmo SVM caiu drasticamente, o que pode apontar um cenário de *overfitting*, para tanto, pelo uso de outras técnicas observou-se que o a técnica *Random Forest Classifier* apresentou resultado mais satisfatórios, todavia ele ainda carece de *tuning*.
- Uma vez criada imagem da solução através do Docker, esta poderia ser hospedada em qualquer máquina virtual dos serviços Cloud, como AWS EC2, Azure Virtual Machine ou Google Compute Engine, todavia, essas soluções costumam pagas, mesmo após o período de *free tier* o que foge o escopo deste projeto, deste modo, uma alternativa mais simplificada encontrada foi hospedar na plataforma Heroku.
- Apesar do Flask oferecer maiores possibilidades, sua complexidade de implementação foi substituída nesta Sprint pelo uso da biblioteca Streamlit, solução sugerida pelo orientador. Com isso foi implementada uma versão inicial do painel de visualização da solução, interface simplificada que ainda carece de melhorias e integração entre as fases de extração, classificação e visualização dos resultados, dentre outras limitações relacionadas pelo plano *free* do Heroku, todavia, por se tratar de um protótipo, fazer uso do Streamlit em substituição ao Flask apresentou vantagem no quesito produtividade.

3. Considerações Finais

3.1 Resultados Finais

O modelo/artefato (MVP) desenvolvido no decorrer das sprints do projeto aplicado satisfaz a pretensão primária de aplicar técnicas de processamento de linguagem natural na classificação de sentimentos/emoções em textos de tweets. De modo que, neste processo de desenvolvimento foi considerado e aplicado técnicas e etapas que contemplaram:

- **Entendimento de negócios:** etapa pautada pelo entendimento e modelagem do problema sob uma ótica de técnicas de *design thinking* para um posterior desenvolvimento baseado em metodologias ágeis;
- **Análise Exploratória dos Dados:** etapa de investigação realizada para compreensão das características dos dados que foram posteriormente usados para o treinamento do modelo;
- **Pré-processamento:** etapa em que se realizou a preparação dos dados através de transformações no texto como: remoção de caracteres especiais, números, pontuações, redução de termos a seus stemmers e remoção de stopwords;
- **Normalização dos Dados:** etapa que contemplou a tokenização dos termos de uma frase, a vetorização dos tokens em uma matriz binária, a transformação de uma matriz binária em uma matriz normalizada que mensura estatisticamente a importância de uma palavra em um determinado texto, e o balanceamento das amostras de cada classe;
- **Modelagem e Avaliação:** etapa em que através da técnica de validação treino e teste (*hold-out*) se avaliou o desempenho de diferentes algoritmos de classificação como: Linear SVM, *Multinomial Naive Bayes*, *Random Forest Classifier*, *KNN*, *SVM*, *Decision Tree*, *Logistic Regression*, *SGD*. Também nesta etapa se avaliou os resultados das técnicas de classificação utilizadas através de métricas como: Acurácia, Precisão, *Recall*, *F1 Score*, e *Auc Roc*, sendo as duas últimas as principais métricas consideradas para determinar um bom desempenho;
- **Sintonização de hiper parâmetros:** etapa em que, a partir dos resultados alcançados na etapa de treinamento dos modelos, foi selecionada a técnica de classificação com melhor desempenho e nesta aplicada a técnica de validação cruzada estratificada em conjunto com técnicas de sintonização de hiper parâmetros, como *Random Search* e *Grid Search*.

- **Exportação do Modelo:** etapa em que o modelo com os hiper parâmetros calibrados foram exportados para um posterior carregamento e testes em um cenário de produção.
- **Ensaio do modelo em produção e hospedagem:** etapa em que, como forma de cumprimento aos objetivos secundários estabelecidos, foram feitos testes de integração do modelo com uma API Web para melhor interação do usuário, para tanto, utilizou-se do *framework* Flask dentro de uma imagem Docker em um cenário de uma classificação individual de tweets e posteriormente houve a escolha pelo *framework* Streamlit como forma de se reduzir a complexidade para se plotar o resultado massivo encontrado por novas classificações.

De forma sucinta, como pontos positivos, pode se citar a possibilidade oferecida durante o desenvolvimento do projeto de se implementar uma solução de ponta a ponta, que contemplasse processos de captura, armazenamento, processamento e visualização, bem como a construção de um modelo cujos resultados de classificação alcançou na fase de treinamento e testes uma taxa de acerto acima dos 80% pelo algoritmo SVC aplicado ao *dataset* “*Portuguese Tweets for Sentiment Analysis*”.

Já como pontos negativos, pode se citar, a queda de desempenho do modelo em testes com os dados do *dataset* “*Brazilian Stock Market Tweets with Emotions*”, de forma que, com este *dataset* o algoritmo *Random Forest Classifier* obteve uma melhor performance, evidenciando que para a adoção de um modelo robusto em um cenário de produção se faz ainda necessário processos de *tuning*. Outro ponto negativo que se deve salientar é que o modelo, na sua construção atual, apresenta uma classificação estática (*snapshot*) de um determinado momento do conjunto dos dados, não contemplando e sua configuração variáveis como tempo e novos dados.

Como principais dificuldades enfrentadas pode se citar a compreensão de detalhes que afetavam o desempenho do modelo como o desbalanceamento das amostras, bem como limitações encontradas na integração entre diferentes tecnologias, como por exemplo a interação entre a API e serviços de *backend*, algo que neste ponto ainda carece de melhorias.

Finalmente, como lições aprendidas, vale pontuar que a simples aplicação de técnicas de processamento de linguagem natural combinadas com técnicas de classificação por si só não garantem o bom desempenho de um modelo, afinal cada problema pode exigir uma solução exclusiva, sendo necessários, além de um bom conhecimento da base de dados e do problema a ser resolvido, diferentes ajustes que podem alterar o desempenho do modelo, e que, modelos ainda que já calibrados em determinado momento, sofrem ações de novos dados, precisando periodicamente reajustes para se manter com um bom desempenho.

3.2 Contribuições

É notório que o uso de técnicas de inteligência artificial na classificação de sentimentos já é um assunto bastante explorado na literatura bem como em tutoriais de processamento de linguagem natural. Todavia, pode-se falar em contribuições, uma vez que, ao direcionar este tipo de classificação a um tema específico, como Ações, se pode alcançar resultados benéficos ao se agregar tais resultados como um novo indicador, visando que este explique ou ratifique a reação do mercado financeiro frente a oscilação dos valores de ações, bem como monitore as emoções expressas por usuários/clientes quanto a uma marca.

3.3 Próximos passos

Para o aprimoramento do artefato desenvolvido e como sugestão pode-se seguir diferentes caminhos como:

- Evoluir a classificação de emoções de um contexto binário para uma classificação de *target multi label*, abrangendo outras emoções que possam estar contidas em um texto de tweet;
- Utilizar técnicas de classificação/previsão de *Deep Learning*, como Redes Neurais Recorrentes, mais especificamente redes com a arquitetura LSTM e comparar os resultados com os algoritmos supervisionados tradicionais.
- Adicionar a variável tempo, e tratar as predições com um problema de modelagem de séries temporais, uma vez que ações do mercado têm em seu comportamento componentes associados como tendência e sazonalidade;
- Desenvolver a solução considerando questões como escalabilidade, utilizando para tanto, tecnologias de Big Data mais robustas para um cenário de produção, idealizando um workflow que contemple a ingestão por streaming, a persistência, o processamento, a visualização e a orquestração de todo o *pipeline*, para tanto recomenda-se fazer uso de tecnologias como o Apache Spark, Apache Kafka, Apache AirFlow, MongoDB, dentre outros.