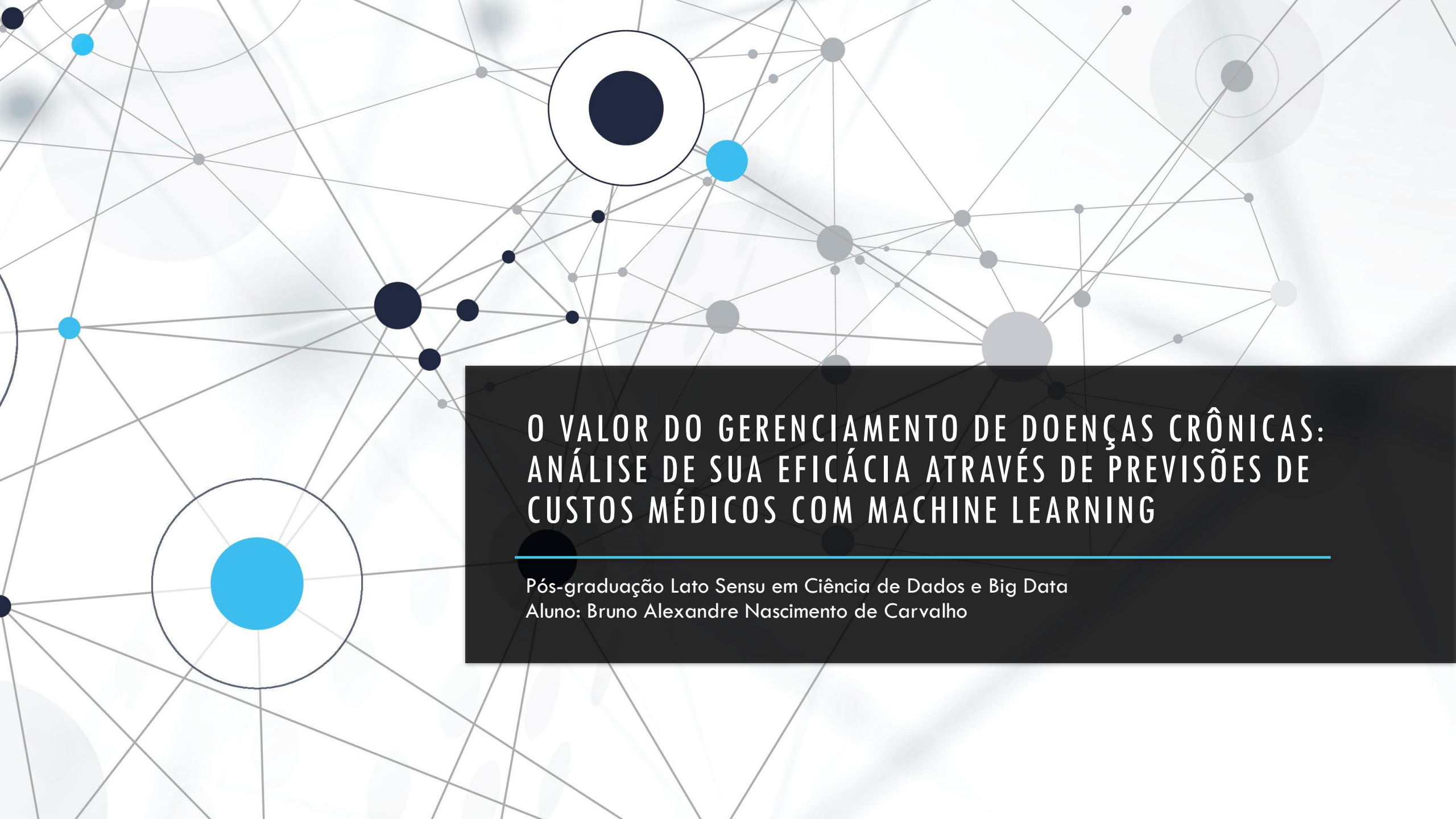


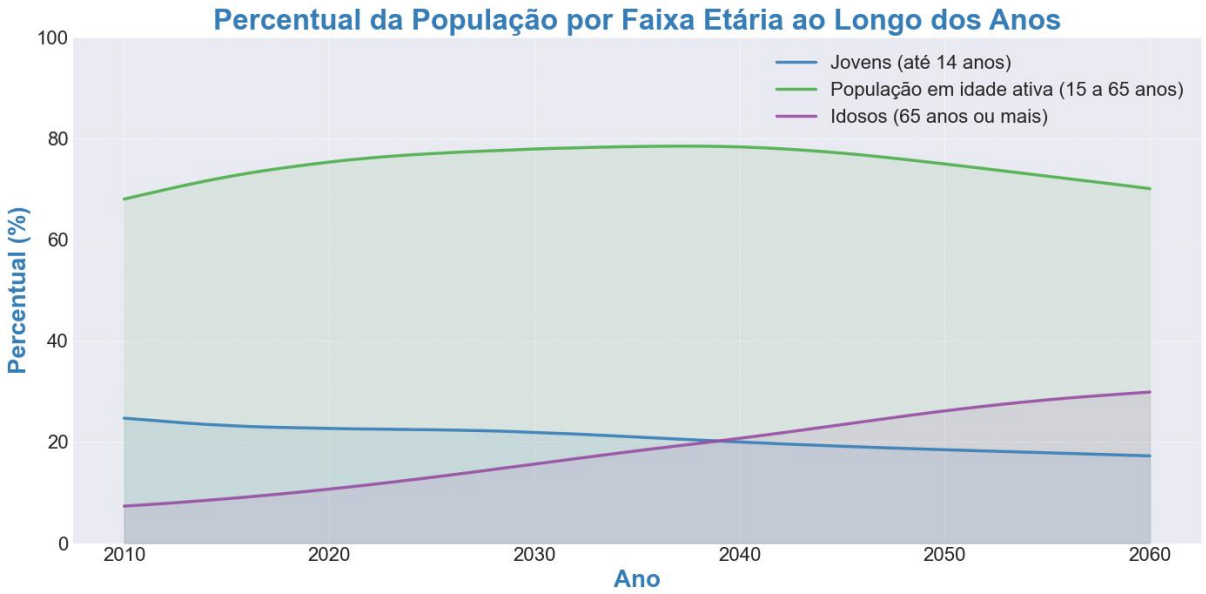
O VALOR DO GERENCIAMENTO DE DOENÇAS CRÔNICAS: ANÁLISE DE SUA EFICÁCIA ATRAVÉS DE PREVISÕES DE CUSTOS MÉDICOS COM MACHINE LEARNING

Pós-graduação Lato Sensu em Ciência de Dados e Big Data
Aluno: Bruno Alexandre Nascimento de Carvalho

1. CONTEXTUALIZAÇÃO

Desafios:

- Envelhecimento da população:



À medida que a população envelhece, as doenças crônico-degenerativas tornam-se mais comuns.

- Crescimento dos custos relacionados à saúde;
- Falta de atuação preventiva.

1. CONTEXTUALIZAÇÃO

Alternativa:

Programas de gerenciamento de doentes crônicos são programas voltados para o gerenciamento de risco em indivíduos com doenças crônicas. São compostos por uma série de ações e comunicações em saúde de forma organizada.

Cenário:

Nesta pesquisa foram estudados os dados de beneficiários acompanhados pelo Programa de Gerenciamento de Doenças Crônicas (PGDC) de um serviço médico assistencial e também os dados de custos médicos da operadora de planos de saúde vinculada a estes beneficiários.

1. OBJETIVO

Aplicar modelos de machine learning para prever o custo médico assistencial dos participantes do PGDC, utilizando dados anteriores ao ingresso no programa, de modo a simular o comportamento da curva de custos na ausência da intervenção do programa e então comparar esse resultado com os valores reais dos primeiros 12 meses de acompanhamento pelo PGDC.

2. COLETA DE DADOS

Banco de dados do **Serviço Assistencial**

Responsável pelo PGDC.

Banco de dados da **operadora de planos de saúde.**

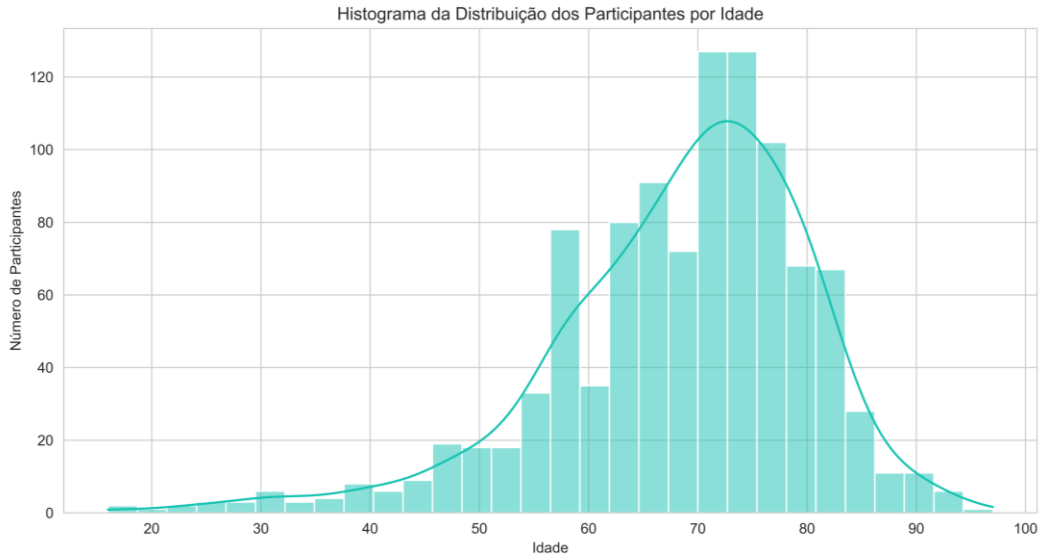
- **Dataset 01**: informações dos participantes do programa (df_participantes_pgdc com **50.367 registros**)
- **Dataset 02**: informações complementares dos participantes (df_complemento_pf **com 2.583 registros**)
- **Dataset 03:** histórico dos custos assistenciais (df_conta_medica com **181.337 registros**)

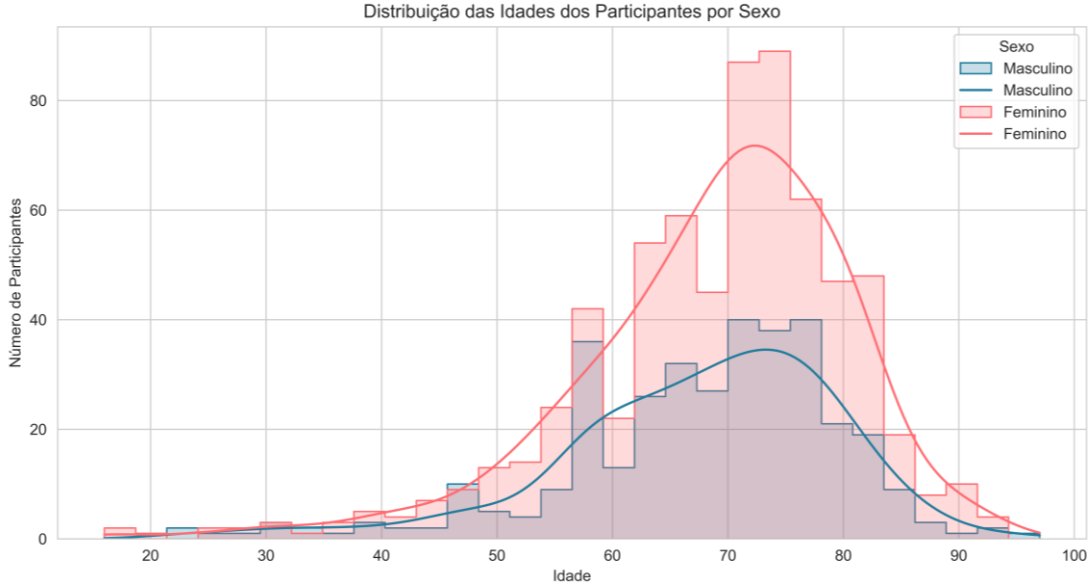
Anonimização dos dados através da conversão das identificações de pessoas físicas em um hash SHA1

3. PROCESSAMENTO/TRATAMENTO DOS DADOS

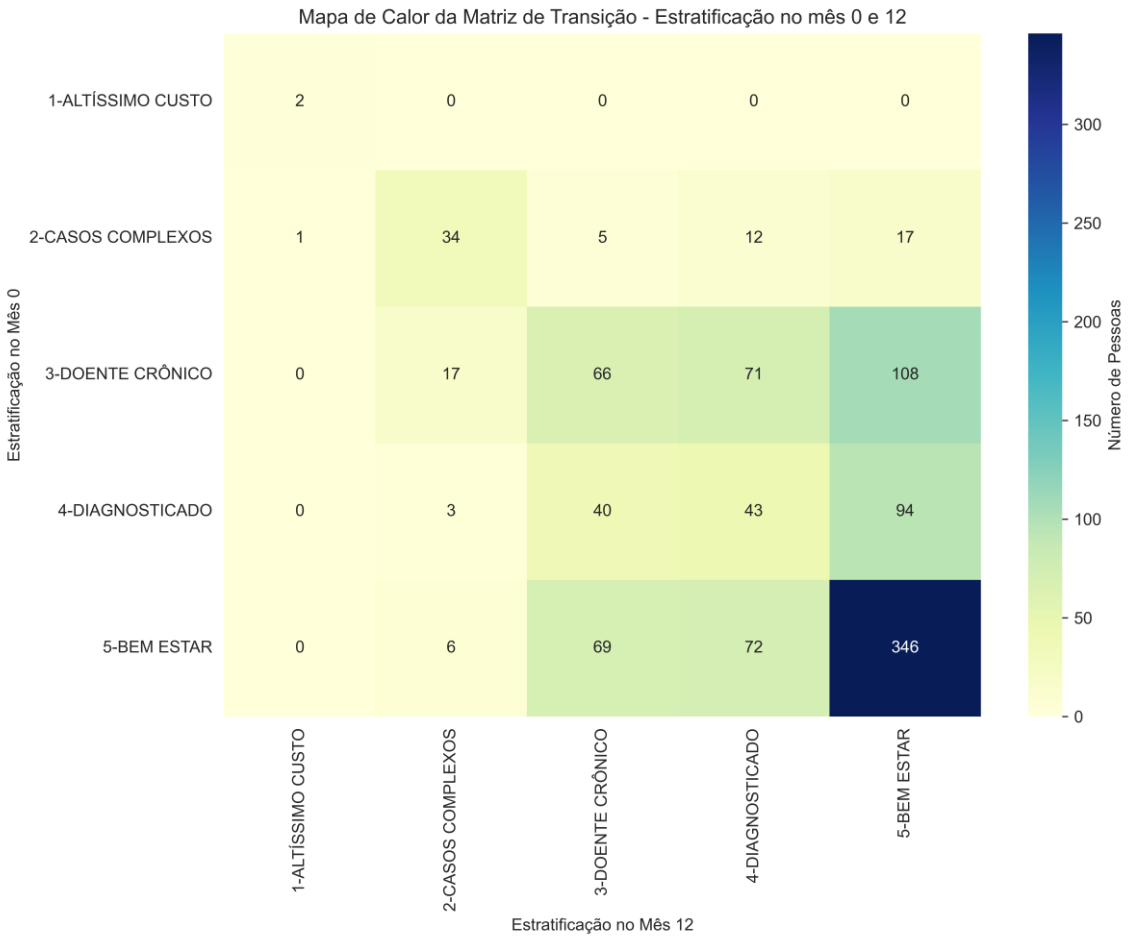
- Remoção de registros com de erros de cadastros;
- Verificação e tratamento de valores duplicados / nulos;
- Remoção das pessoas com menos de 12 meses de acompanhamento pelo programa;
- Agrupamento dos dados;
- Enriquecimento dos dados com o dataset de complemento de pessoa física (sexo e data de nascimento);
- Enriquecimento dos dados com o dataset de custos assistenciais;
- Remoção de participantes sem informação de custo assistencial;
- Criação do atributo QT_MES_REFERENCIA que foi utilizada para definir o período de ocorrência do custo médico em relação ao início da participação no programa;
- Agrupamento dos registros por mês de ocorrência do custo médico;
- Cálculo da média mensal do custo assistencial dos participantes do programa.

4. ANÁLISE E EXPLORAÇÃO DOS DADOS

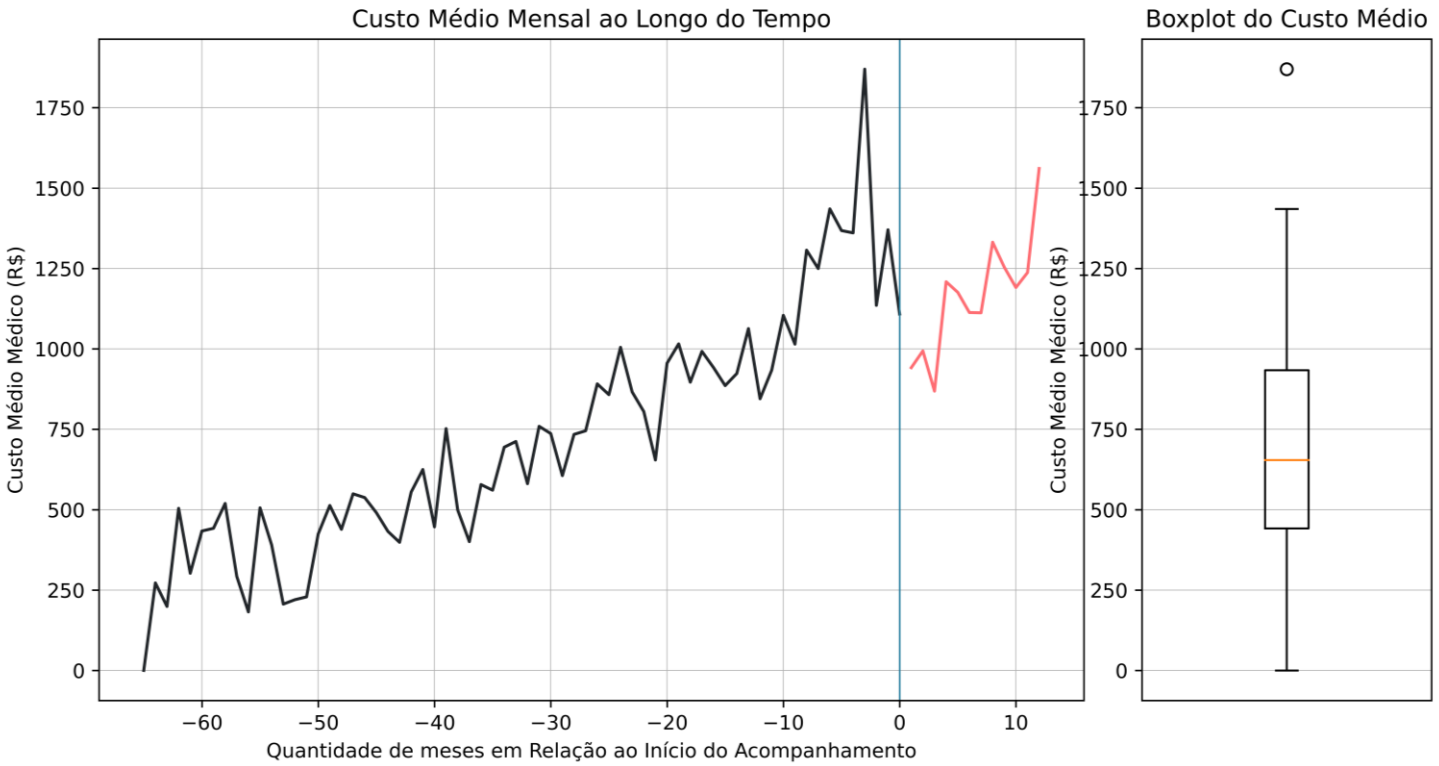




4. ANÁLISE E EXPLORAÇÃO DOS DADOS

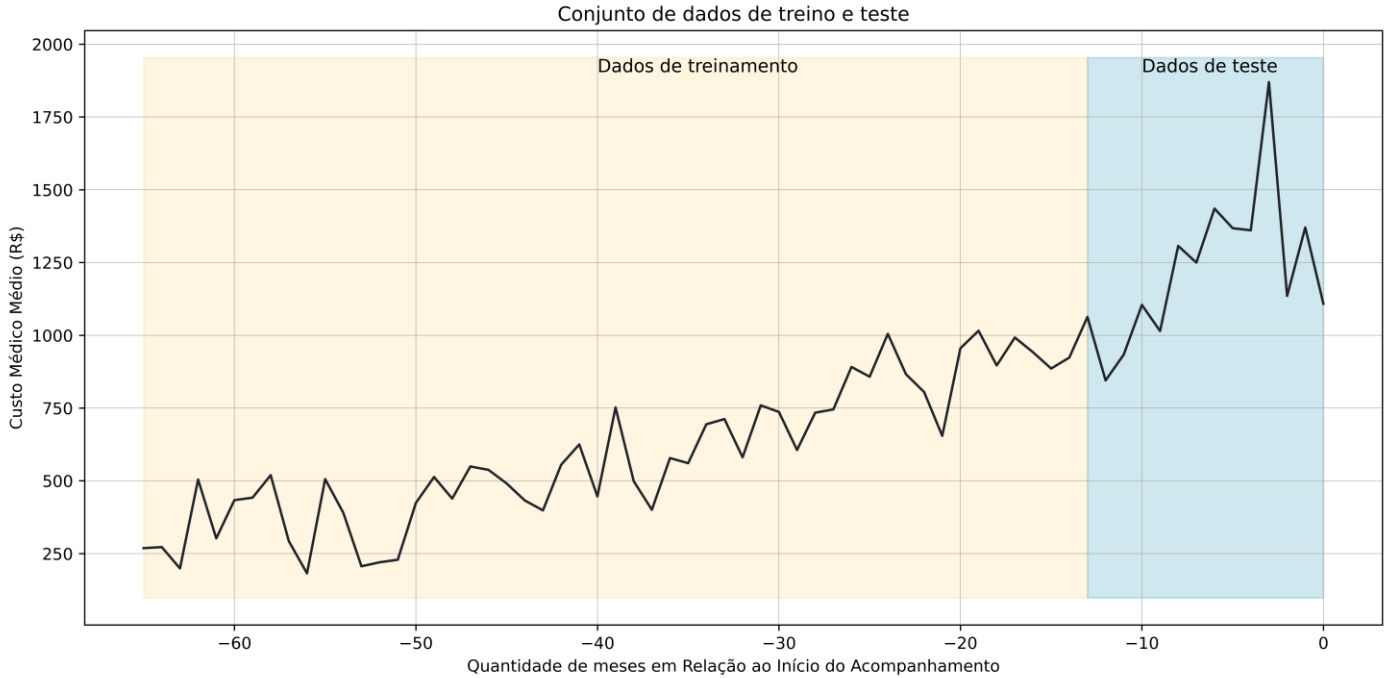


4. ANÁLISE E EXPLORAÇÃO DOS DADOS



5. MODELOS DE MACHINE LEARNING

- Holt-Winters
- ARIMA
- MLPRegression (scikit-learn) scikit-learn



5. MODELOS DE MACHINE LEARNING

- Holt-Winters:
  - Busca dos melhores parâmetros pela comparação do AIC.

```python
# Função para definir os parâmetros do modelo HW com base no AIC
def holt_winters_aic(trend, seasonal, seasonal_periods, dados):
    try:
        modelo = ExponentialSmoothing(dados, trend=trend, seasonal=seasonal, seasonal_periods=seasonal_periods)
        fit_modelo = modelo.fit()
        aic = fit_modelo.aic
        return aic
    except:
        return np.inf

# Parâmetros possíveis para o modelo Holt-Winters
trends = [None, 'add', 'mul']
seasonals = [None, 'add', 'mul']
seasonal_periods = list(range(1, 25))

# Busca otimizada dos melhores parâmetros usando AIC
min_aic = np.inf
melhores_params_aic = None
for trend in trends:
    for seasonal in seasonals:
        for seasonal_period in seasonal_periods:
            aic = holt_winters_aic(trend, seasonal, seasonal_period, dados_treino['VL_CUSTO_MEDIO'])
            if aic < min_aic:
                min_aic = aic
                melhores_params_aic = (trend, seasonal, seasonal_period)

print(f"Parâmetros otimizados para menor AIC: {melhores_params_aic}")
print(f"Menor AIC: {min_aic}")
```

✓ 11.2s

5. MODELOS DE MACHINE LEARNING

- ARIMA:
  - Parametrização manual
  - AUTOARIMA

```
arima_auto = auto_arima(dados_treino['VL_CUSTO_MEDIO'], start_p=1, start_q=1,
                        max_p=6, max_q=6, m=12, start_P=0, seasonal=True,
                        d=1, D=1, trace=True, error_action='ignore',
                        suppress_warnings=True, stepwise=True)
```

✓ 11.7s

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=inf, Time=0.26 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=544.562, Time=0.01 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=530.617, Time=0.08 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=0.15 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=537.793, Time=0.03 sec
 ARIMA(1,1,0)(2,1,0)[12]             : AIC=528.685, Time=0.19 sec
 ARIMA(1,1,0)(2,1,1)[12]             : AIC=530.648, Time=0.74 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=inf, Time=0.33 sec
 ARIMA(0,1,0)(2,1,0)[12]             : AIC=532.873, Time=0.24 sec
 ARIMA(2,1,0)(2,1,0)[12]             : AIC=523.591, Time=0.39 sec
 ARIMA(2,1,0)(1,1,0)[12]             : AIC=528.009, Time=0.14 sec
 ARIMA(2,1,0)(2,1,1)[12]             : AIC=525.519, Time=0.61 sec
 ARIMA(2,1,0)(1,1,1)[12]             : AIC=inf, Time=1.06 sec
 ARIMA(3,1,0)(2,1,0)[12]             : AIC=525.108, Time=0.43 sec
 ARIMA(2,1,1)(2,1,0)[12]             : AIC=522.287, Time=0.52 sec
 ARIMA(2,1,1)(1,1,0)[12]             : AIC=525.008, Time=0.29 sec
 ARIMA(2,1,1)(2,1,1)[12]             : AIC=524.418, Time=1.13 sec
 ARIMA(2,1,1)(1,1,1)[12]             : AIC=inf, Time=0.67 sec
 ARIMA(1,1,1)(2,1,0)[12]             : AIC=520.756, Time=0.31 sec
 ARIMA(1,1,1)(1,1,0)[12]             : AIC=523.084, Time=0.19 sec
 ARIMA(1,1,1)(2,1,1)[12]             : AIC=inf, Time=0.64 sec
 ARIMA(1,1,1)(1,1,1)[12]             : AIC=inf, Time=0.42 sec
 ARIMA(0,1,1)(2,1,0)[12]             : AIC=519.088, Time=0.33 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=521.337, Time=0.11 sec
...
 ARIMA(0,1,1)(2,1,0)[12] intercept   : AIC=inf, Time=0.28 sec

Best model:  ARIMA(0,1,1)(2,1,0)[12]
Total fit time: 11.750 seconds
```

5. MODELOS DE MACHINE LEARNING

- MLPRegressor:
  - Transformação do conjunto de dados (3 lags)
  - GridSearch

```
from sklearn.model_selection import GridSearchCV

# Definindo os hiperparâmetros para teste
parametros = {
    'hidden_layer_sizes': [(1,), (2,), (3,), (4,), (5,), (6,), (7,), (8,),
                           (9,), (10,), (11,), (12,), (13,), (14,), (15,), (16,)],
    'activation': ['logistic', 'relu', 'tanh'],
    'solver': ['adam','lbfgs', 'sgd',],
    'max_iter': [500, 5000, 20000],
    'learning_rate_init': [0.001, 0.01, 0.1]
}

# Configurando o GridSearchCV
grid_search = GridSearchCV(modelo_mlpregressor, parametros, cv=n_series, verbose=1, n_jobs=-1)

# Realizando a busca pelos melhores hiperparâmetros
grid_search.fit(X, y)

# Resultados
melhores_parametros = grid_search.best_params_

melhores_parametros
```

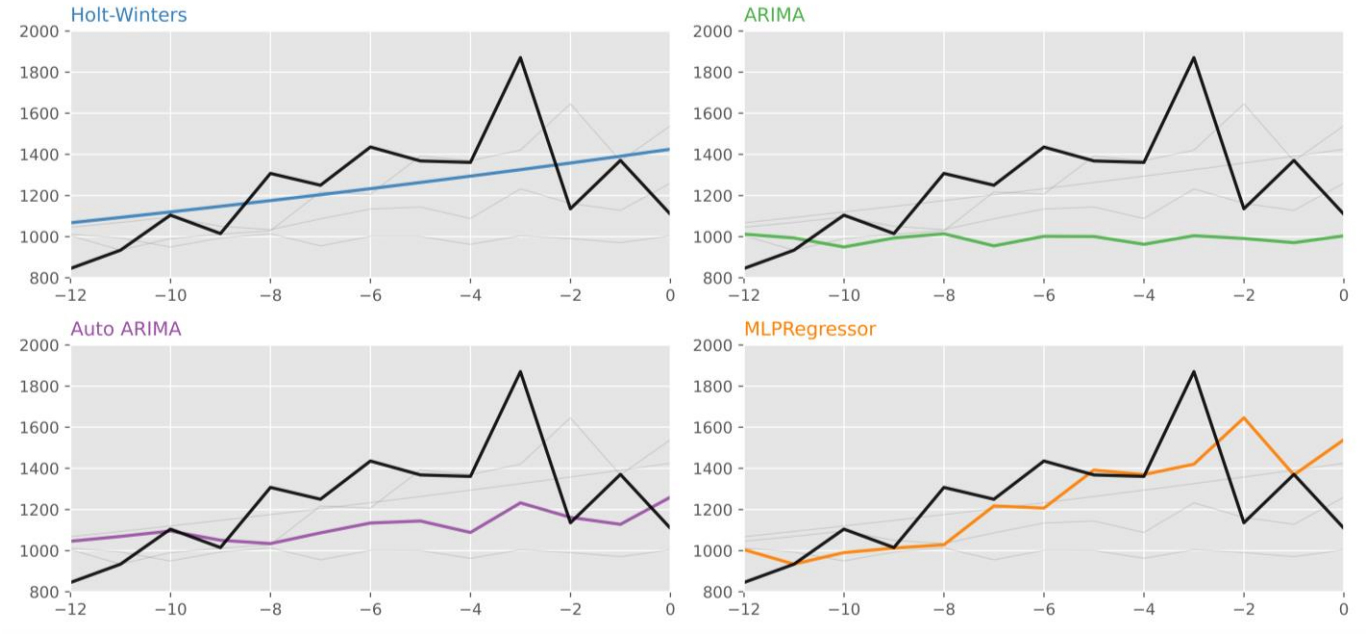
✓ 57m 37.4s

```
Fitting 5 folds for each of 1296 candidates, totalling 6480 fits
c:\Users\bruno\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:692: ConvergenceWarning:

Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.


{'activation': 'relu',
 'hidden_layer_sizes': (9,),
 'learning_rate_init': 0.1,
 'max_iter': 500,
 'solver': 'adam'}
```

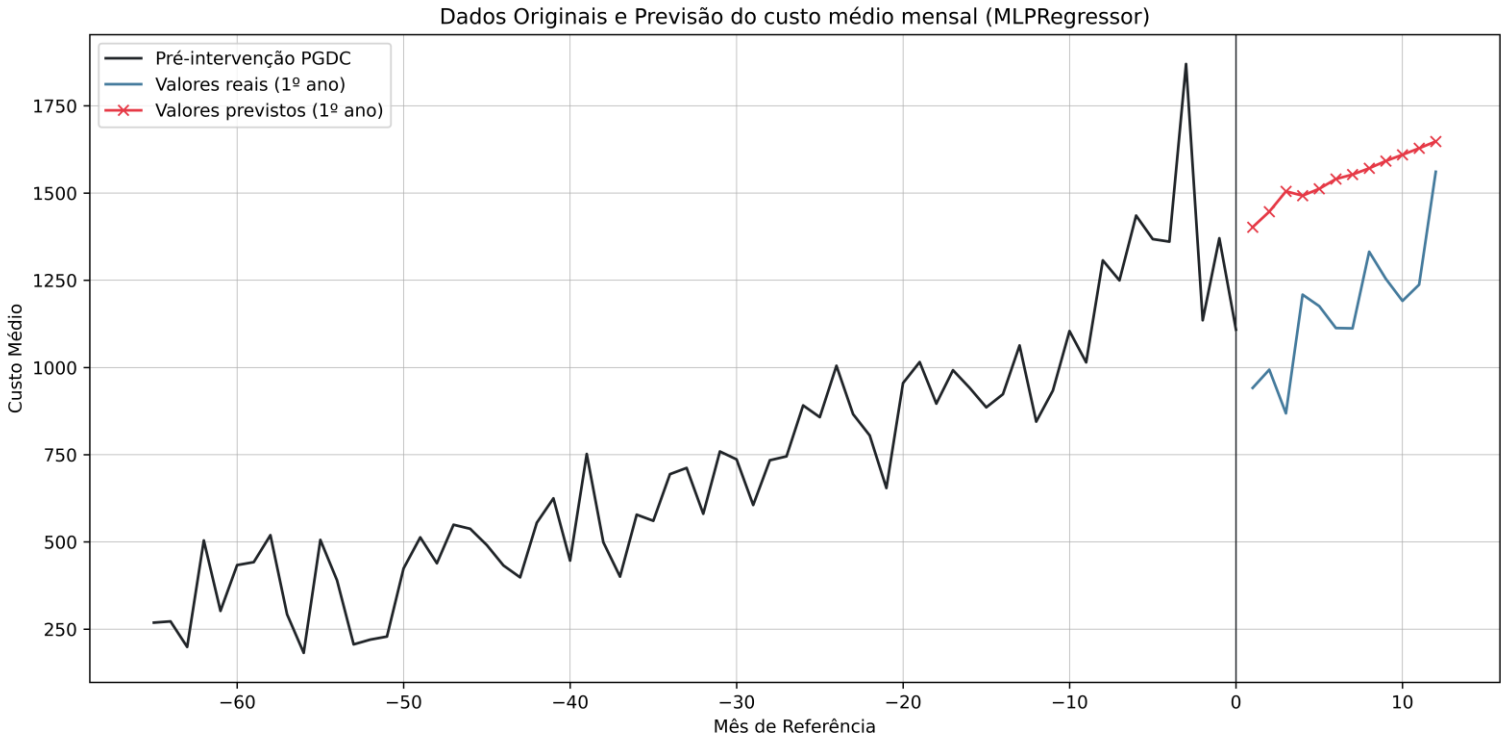
6. INTERPRETAÇÃO DOS RESULTADOS

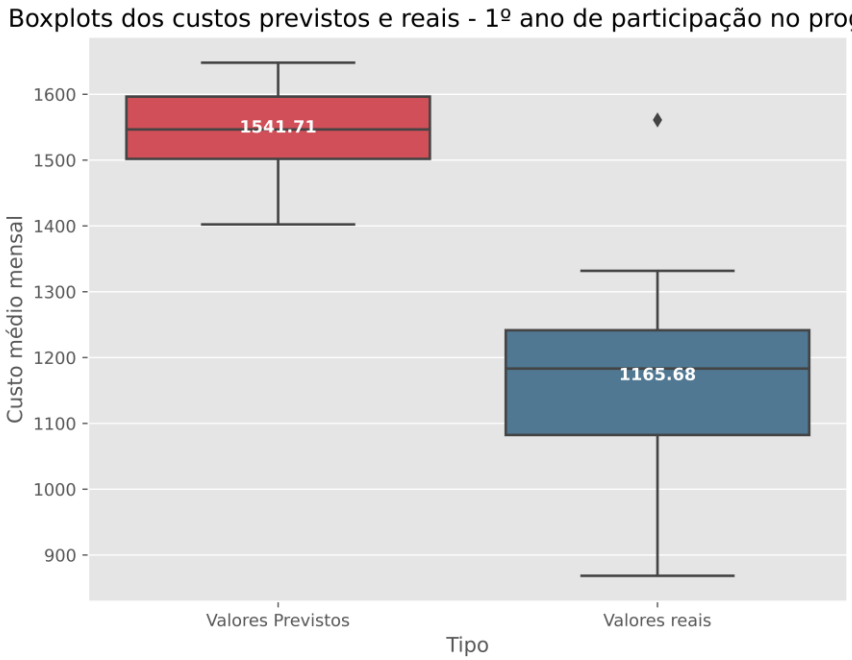
- Métricas avaliadas:

	Holt-Winters	ARIMA(2,1,5)	Auto ARIMA	MLPRegressor
MAE	168.090000	285.050000	205.610000	172.490000
MSE	47301.030000	127088.610000	66599.030000	62947.310000
RMSE	217.490000	356.490000	258.070000	250.890000
MAPE	0.140000	0.210000	0.160000	0.140000
AIC	507.325168	659.993434	519.087737	235.650694



6. INTERPRETAÇÃO DOS RESULTADOS





6. INTERPRETAÇÃO DOS RESULTADOS

- Teste de Shapiro-Wilk: verificar a normalidade;
- Teste de Levene: avaliar a homocedasticidade;
- Teste t de Student: diferença entre ambos os conjuntos é estatisticamente significativa.

```
from scipy.stats import shapiro, levene, ttest_ind

# Teste de Normalidade para ambos os conjuntos (Shapiro-Wilk)
p_valor_real = shapiro(df_final[df_final["TIPO"] == "Valores reais"]["VL_CUSTO_MEDIO"])
p_valor_previsto = shapiro(df_final[df_final["TIPO"] == "Valores Previstos"]["VL_CUSTO_MEDIO"])

# Teste de Homocedasticidade (Levene)
p_valor_levene = levene(df_final[df_final["TIPO"] == "Valores reais"]["VL_CUSTO_MEDIO"],
                        df_final[df_final["TIPO"] == "Valores Previstos"]["VL_CUSTO_MEDIO"])

p_valor_real, p_valor_previsto, p_valor_levene
```

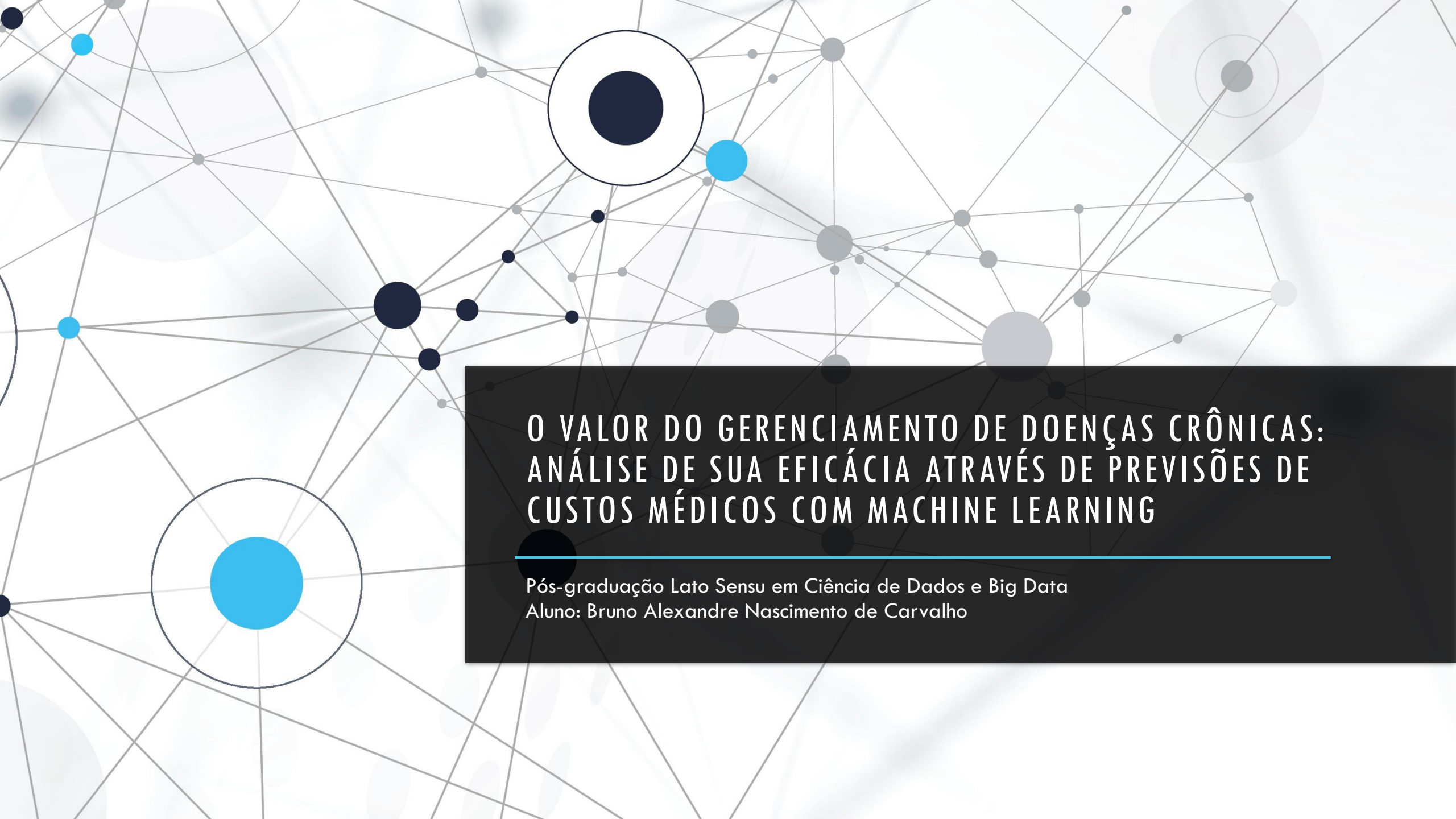
✓ 0.0s

```
(ShapiroResult(statistic=0.9619141221046448, pvalue=0.8107696771621704),
 ShapiroResult(statistic=0.9755776524543762, pvalue=0.9596397876739502),
 LeveneResult(statistic=3.7340573701017963, pvalue=0.06629110953613954))
```

```
# Teste t de Student para amostras independentes
t_stat, p_valor_ttest = ttest_ind(df_final[df_final["TIPO"] == "Valores reais"]["VL_CUSTO_MEDIO"],
                                  df_final[df_final["TIPO"] == "Valores Previstos"]["VL_CUSTO_MEDIO"])

t_stat, p_valor_ttest
```

✓ 0.0s

```
(-6.563774450940586, 1.3336525303813554e-06)
```



O VALOR DO GERENCIAMENTO DE DOENÇAS CRÔNICAS: ANÁLISE DE SUA EFICÁCIA ATRAVÉS DE PREVISÕES DE CUSTOS MÉDICOS COM MACHINE LEARNING

Pós-graduação Lato Sensu em Ciência de Dados e Big Data
Aluno: Bruno Alexandre Nascimento de Carvalho