Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas

TÓPICOS EM SISTEMAS INTELIGENTES

Engenharia de Computação - 8º Período

Computação Financeira

LUAN ANTONIO DOMINGOS VENANCIO

18 de Outubro de 2023

1. Metodologia

O framework para criação de estratégias de investimento automatizado foi desenvolvido utilizando Aprendizado de Máquina para tomada de decisão de compra e venda de ações da empresa Localiza (RENT3). As etapas de construção da base de dados e simulação do retorno foram implementadas no mesmo projeto Java feito na IDE Netbeans, as etapas um e três, respectivamente.

A etapa dois, que tem como objetivo, construção do modelo de aprendizado e a geração das saídas de teste, a linguagem escolhida foi Python, e todo o código foi feito na ferramenta Google Colab. As bibliotecas utilizadas foram:

- Pandas: Estruturas de dados e operações para manipular e analisar dados em tabelas;
- NumPy: Estruturas de dados e operações para trabalhar com matrizes e arrays;
- Scikit-learn: Algoritmos de aprendizado de máquina;
- ta: Análise técnica para conjuntos de dados de séries temporais financeiras.

2. Algoritmos *Baseline*

2.1 *Baseline* de classificação

Para realização do algoritmo de baseline de classificação, foi verificada a distribuição da classe majoritária no conjunto de treinamento através do método value_counts().

```
y_pred_base_classifier = pd.DataFrame(y_test)
y_pred_base_classifier.value_counts()
```

```
Output
1         211
0         206
dtype: int64
```

O valor que apareceu mais vezes foi o 1. Portanto, todos os valores foram alterados para 1.

```
y_pred_base_classifier['Output'] = 1
y_pred_base_classifier
```

	Output
2907	1
2908	1
2909	1
2910	1
2911	1
...	...
3319	1
3320	1
3321	1
3322	1
3323	1

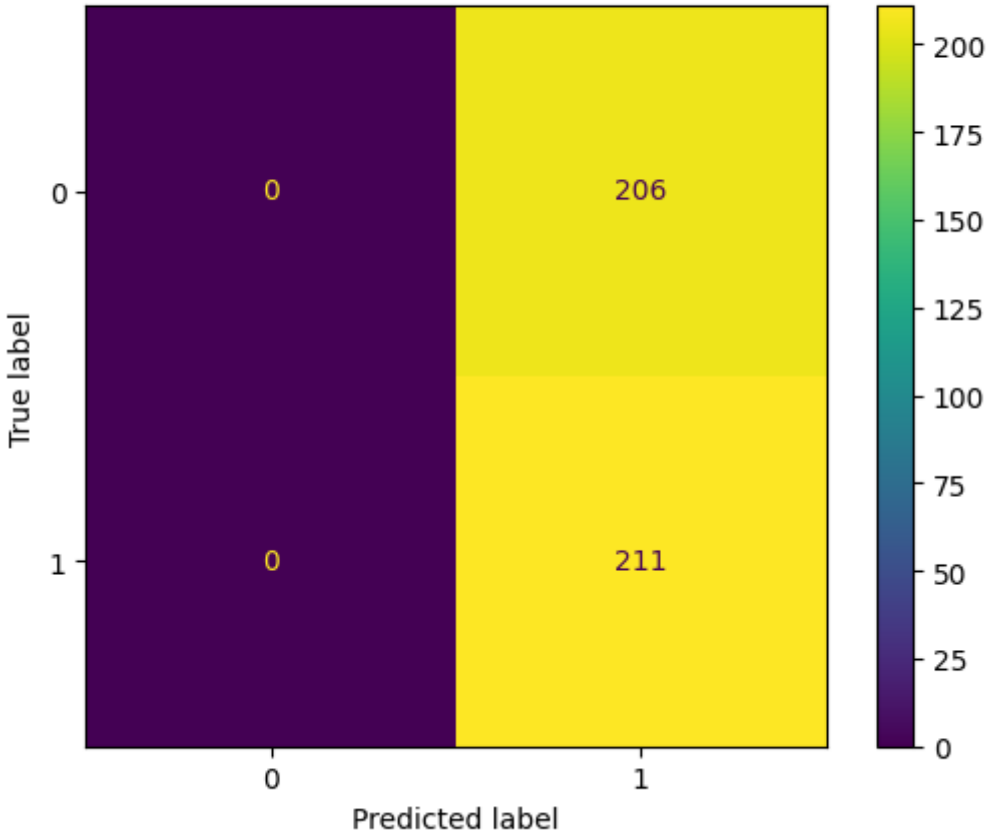
417 rows × 1 columns

Para avaliar os modelos de classificação foi criado uma função chamada evaluate_classification, que retorna as métricas de Acurácia, Precisão, Recall e AUC:

```python
from sklearn.metrics import recall_score, precision_score, accuracy_score, roc_auc_score, confusion_matrix, ConfusionMatrixDisplay

def evaluate_classification(y_test, y_pred):
  cm = confusion_matrix(y_test, y_pred)

  ConfusionMatrixDisplay(confusion_matrix=cm).plot();

  accuracy = accuracy_score(y_test, y_pred)
  precision = precision_score(y_test, y_pred)
  recall = recall_score(y_test, y_pred)
  AUC = roc_auc_score(y_test, y_pred)

  print("Accuracy:", accuracy)
  print("Precision:", precision)
  print("Recall:", recall)
  print("AUC:", AUC)
```

O resultado do baseline de classificação foi:

```
Accuracy: 0.5059952038369304
Precision: 0.5059952038369304
Recall: 1.0
AUC: 0.5
```



O Baseline tem um AUC de 0.5 e um Recall de 1 que bate com o esperado.

2.2 *Baseline* de regressão

Para realização do algoritmo de *baseline* de regressão, equivalente à previsão do retorno do dia anterior como previsão do dia seguinte, através do método shift().

	Return
2907	-0.29
2908	1.47
2909	-0.19
2910	0.42
2911	0.97
...	...
3319	2.52
3320	0.15
3321	-0.78
3322	0.17
3323	0.22

417 rows × 1 columns

```
y_pred_base_regressor = pd.DataFrame(y_test).shift(1, fill_value=0)
y_pred_base_regressor
```

	Return
2907	0.00
2908	-0.29
2909	1.47
2910	-0.19
2911	0.42
...	...
3319	0.87
3320	2.52
3321	0.15
3322	-0.78
3323	0.17

417 rows × 1 columns

Para avaliar os modelos de regressão foi criado uma função chamada evaluate_regressor, que retorna as métricas de RMSE e MAE:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

def evaluate_regressor(y_test, y_pred):
  test_set_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
  print('RMSE = ', test_set_rmse)

  mae = mean_absolute_error(y_test, y_pred)
  print("MAE = ", mae)
```

O resultado do baseline de regressão foi:

```
RMSE =  1.8736469778487927
MAE =  1.4565947242206225
```

3. Algoritmo de Aprendizado de Máquinas

2.1 Engenharia de Atributos

O objetivo da engenharia de atributos é a de melhorar a qualidade e a relevância dos dados para o modelo, o que pode resultar em um melhor desempenho. Foram criados diversos atributos utilizando:

- Lag Features: Calculadas a partir de valores anteriores de uma série temporal;
- Séries Temporais: Calculadas a partir de uma série temporal;
- Indicadores Técnicos: Usados para analisar séries temporais de preços de ações, alguns exemplos são: Indicadores de momento, Indicadores de tendência e Indicadores de volume.

```python
# Close Lag Feature
df['Close_Lag_1'] = close.shift(1)

# Volume difference
df['Volume_diff'] = volume.diff()

# Volume Weighted Average Price
df['vwap'] = (close * volume).cumsum() / volume.cumsum()

# Volume Percentage Change
df['volume_percentage_change'] = volume.pct_change()

# Daily Return
df['daily_Return'] = close.pct_change() * 100
```

```python
interators = [2, 3, 4, 5, 10]

for interator in interators:

    # Lag Feature
    close_lag_column = f"Close_Lag_{interator}"
    df[close_lag_column] = close.shift(interator)

    # Rolling Avg
    rolling_averages = df.rolling(interator).mean()
    close_rolling_mean_column = f"Close_Rolling_Mean_{interator}"
    df[close_rolling_mean_column] = rolling_averages["Close"]

    # Close Ratio
    close_ratio_column = f"Close_Ratio_{interator}"
    df[close_ratio_column] = close / rolling_averages["Close"]

    # Rolling Standard Deviation
    rolling_std = df.rolling(interator).std()
    close_rolling_std_column = f"Close_Rolling_Std_{interator}"
    df[close_rolling_std_column] = rolling_std["Close"]

    # Expanding Avg
    expanding_averages = df.expanding(interator).mean()
    close_expanding_mean_column = f"Close_Expanding_Mean_{interator}"
    df[close_expanding_mean_column] = expanding_averages["Close"]

    # Expanding Standard Deviation
    expanding_std = df.expanding(interator).std()
    close_expanding_std_column = f"Close_Expanding_Mean_{interator}"
    df[close_expanding_std_column] = expanding_std["Close"]

    # Exponential Moving Avg
    exponential_mov_avg = df.ewm(span=interator, adjust=False).mean()
    exponential_moving_avg_column = f"Exponential_Moving_Avg_{interator}"
    df[exponential_moving_avg_column] = exponential_mov_avg["Close"]

    # Simple Moving Average
    sma = df.rolling(interator).mean()
    sma_mean_column = f"sma_{interator}"
    df[sma_mean_column] = sma["Close"]
```

```
# Moving Average Convergence Divergence (MACD)
df['MACD'] = ta.trend.macd_diff(df['Close'])

# Relative Strength Index (RSI)
df['RSI_5'] = ta.momentum.rsi(df['Close'], window=5)
df['RSI_14'] = ta.momentum.rsi(df['Close'], window=14)

# Money Flow Multiplier: [(Close - Low) - (High - Close)] /(High - Low)
money_flow_mult = ((close - low) - (high - close)) /(high - low)
df['money_flow_mult'] = money_flow_mult

# Money Flow Volume: Money Flow Multiplier x Volume for the Period
money_flow_volume = money_flow_mult * volume
df['money_flow_volume'] = money_flow_volume

# ADL(ADI): Previous ADL + Current Period's Money Flow Volume
adi = money_flow_volume.cumsum()
df['adi'] = adi

# Chaikin Money Flow
cmf = money_flow_volume.rolling(20).sum() / volume.rolling(20).sum()
df['chaikin_money_flow'] = cmf
```

Depois todos os valores faltando foram removidos.

```
df = df.replace([np.inf, -np.inf], np.nan)
df = df.dropna()

df.head()
```

2.2 Classificação

O modelo de classificação utiliza o Random Forest, um algoritmo de aprendizado de máquina que pode ser usado para prever se o preço de uma ação irá subir ou cair. O modelo é treinado em um conjunto de dados histórico de preços de ações, e usa essa informação para aprender como identificar padrões.

O Random Forest funciona treinando várias árvores de decisão independentes, e depois combinando os resultados dessas árvores para fazer uma previsão. Cada árvore de decisão é treinada em um subconjunto aleatório dos dados, o que ajuda a reduzir o viés e a variância do modelo.

2.2.1 *Hold out*

A técnica de validação *Hold out* consiste em dividir o conjunto de dados em dois subconjuntos: um subconjunto de treinamento e um subconjunto de teste. O modelo é treinado no subconjunto de treinamento e avaliado no subconjunto de teste.

O conjunto de dados foi dividido em dois subconjuntos, o de treinamento com dados até 2022, e o de teste com dados a partir de 2022:

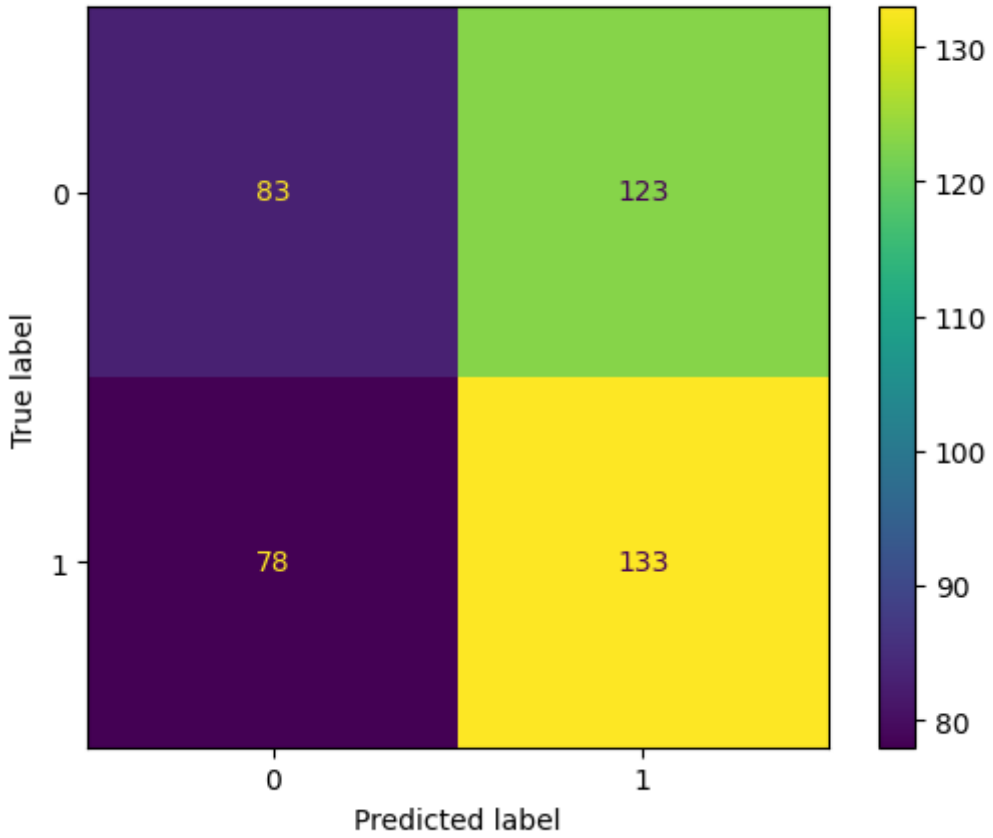
```
y_train = df.loc[df['year'] < 2022, 'Output']
X_train = df.loc[df['year']  < 2022].drop(['Output', 'Return', 'Date'], axis="columns")

y_test = df.loc[df['year']  >= 2022, 'Output']
X_test = df.loc[df['year']  >= 2022].drop(['Output', 'Return', 'Date'], axis="columns")
```

Treinamento do modelo:

```
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import VarianceThreshold
from sklearn.ensemble import RandomForestClassifier

# Aplica Random Forest Classification para o conjunto de treinamento
classifier = RandomForestClassifier(random_state=1111)
classifier.fit(X_train, y_train)

# Predição com os o conjunto de teste
y_pred = classifier.predict(X_test)
```

Avaliação do modelo:

```
Accuracy: 0.5179856115107914
Precision: 0.51953125
Recall: 0.6303317535545023
AUC: 0.5166221874568628
```



A técnica *hold out* apresentou uma pequena melhora se comparada com o baseline.

2.2.2 Janela Deslizante

A técnica da janela deslizante consiste em dividir o conjunto de dados em uma série de janelas, e então treinar e avaliar o modelo em cada janela.

Para implementar a janela deslizante foi criado uma função chamada expanding_window, que recebe o modelo que será utilizado no treinamento (Random Forest ou MLP), X_train e y_train que será utilizado para o treinamento do modelo, e o window_size que corresponde ao tamanho da janela (Mensal, Semanal ou diária).

```python
from sklearn.model_selection import TimeSeriesSplit
from sklearn.preprocessing import StandardScaler

def expanding_window(model, X_train, y_train, window_size):

    pred = []
    actuals = []

    size = len(df.loc[df['year'] < 2022].index)
    size_max =  len(df)

    train_starts = range(size, size_max, window_size)
    i = 0

    for train_start in train_starts:

        X_train_window, X_test_window = X_train[:train_start], X_train[train_start : train_start + window_size]
        y_train_window, y_test_window = y_train[:train_start], y_train[train_start : train_start + window_size]

        print(f"Fold: {i}")
        print(f"Train: index={X_train[:train_start].index} - Size: {X_train[:train_start].size}")
        print(f"Test:  index={y_train[train_start : train_start + window_size].index} - Size: {y_train[train_start : train_start + window_size].size} \n")

        model.fit(X_train_window, y_train_window)
        y_pred_window = model.predict(X_test_window)

        i+=1
        pred.extend(y_pred_window)
        actuals.extend(y_test_window)

    return np.array(pred), np.array(actuals)
```

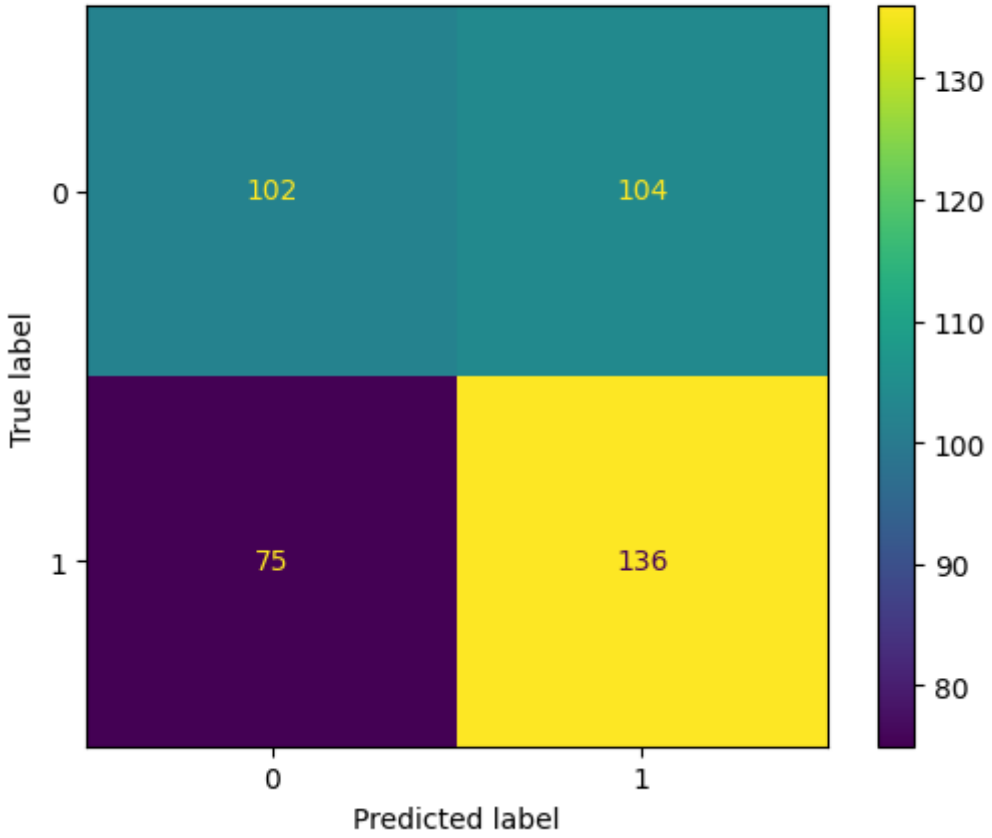
2.2.3 Janela Deslizante Mensal

O conjunto de dados é dividido em uma série de janelas de tamanhos iguais, onde cada janela representa um mês (22 dias). Implementação da Janela deslizante mensal:

```python
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(random_state=1111)
y_pred_window, y_test_window = expanding_window(classifier, X, y, window_size=22)
```

Avaliação do modelo:

```
Accuracy: 0.5707434052757794
Precision: 0.5666666666666667
Recall: 0.6445497630331753
AUC: 0.5698476970505683
```



A técnica de Janela Deslizante Mensal apresentou uma melhora significativa se comparada com o baseline e *hold out*.

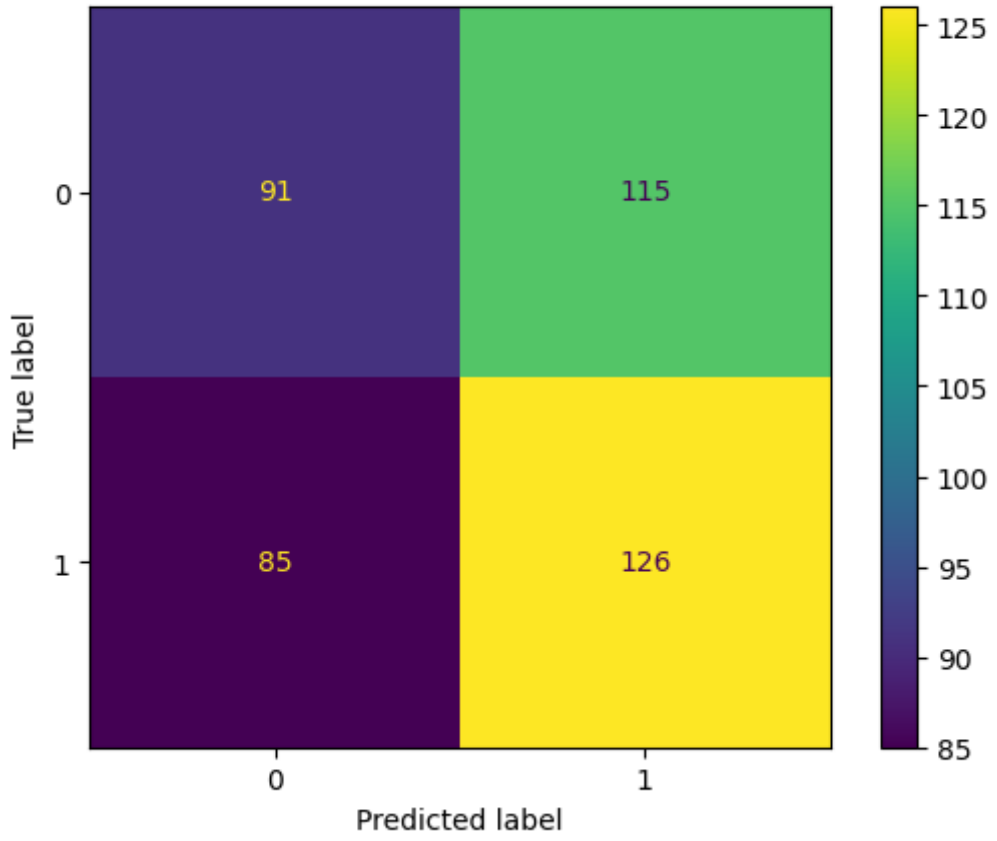
2.2.4 Janela Deslizante Semanal

O conjunto de dados é dividido em uma série de janelas de tamanhos iguais, onde cada janela representa uma semana (5 dias). Implementação da Janela deslizante semanal:

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(random_state=1111)
y_pred_window, y_test_window = expanding_window(classifier, X, y, window_size=5)
```

Avaliação do modelo:

```
Accuracy: 0.5203836930455635
Precision: 0.5228215767634855
Recall: 0.5971563981042654
AUC: 0.5194519854598997
```



A técnica de Janela Deslizante Semanal piorou significativamente se comparado com a Janela Deslizante de um mês.

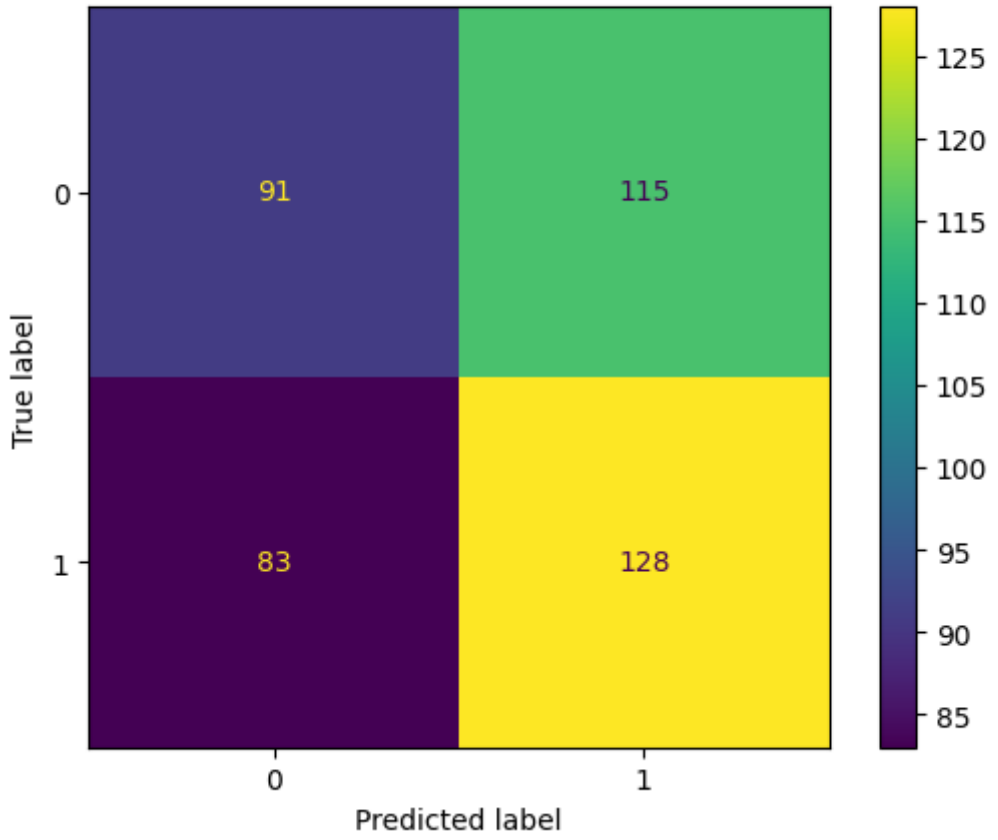
2.2.5 Janela Deslizante Diária

O conjunto de dados é dividido em uma série de janelas de tamanhos iguais, onde cada janela representa um dia. Implementação da Janela deslizante diária:

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(random_state=1111)
y_pred_window, y_test_window = expanding_window(classifier, X, y, window_size=1)
```

Avaliação do modelo:

```
Accuracy: 0.5251798561151079
Precision: 0.5267489711934157
Recall: 0.6066350710900474
AUC: 0.5241913219527907
```



A técnica de Janela Deslizante Diária melhorou se comparada com a Janela Deslizante Mensal, porém, ainda é pior que os resultados da Janela Deslizante Mensal.

2.3 Regressão

O modelo de regressão é utilizado para prever um valor numérico, como o retorno de uma ação. Para isso, é utilizado o MLP, ou redes neurais multicamadas, que são um tipo de modelo de aprendizado de máquina, compostos por um conjunto de camadas de neurônios, cada uma com um número específico de neurônios. As camadas são interconectadas por ligações, que são representadas por pesos. O valor alvo é atribuído a uma camada de saída, que é conectada a todas as camadas anteriores.

2.3.1 *Hold out*

O conjunto de dados foi dividido em dois subconjuntos:

```
y_train = df.loc[df['year'] < 2022, 'Return']
X_train = df.loc[df['year']  < 2022].drop(['Output', 'Return', 'Date'], axis="columns")

y_test = df.loc[df['year'] >= 2022, 'Return']
X_test = df.loc[df['year'] >= 2022].drop(['Output', 'Return', 'Date'], axis="columns")
```

Treinamento do modelo:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.neural_network import MLPRegressor

# Aplica MLP para o conjunto de treinamento
regressor = MLPRegressor(random_state=1111)
regressor.fit(X_train, y_train)

# Predição com os o conjunto de teste
y_pred = regressor.predict(X_test)
```

Avaliação do modelo:

```
RMSE =  1.3548437627944685
MAE =  1.053642392057289
```

A técnica *hold out* apresenta valores altos de RMSE e MAE, porém, menores que o baseline.

2.3.2 Janela Deslizante

O código para implementar a janela deslizante é o mesmo do algoritmo de classificação, a diferença é que agora o modelo que função recebe não é mais o de

classificação, e sim, o de regressão. Conjunto de dados dividido em dois subconjuntos:

```
y = df['Return']
X = df.drop(['Output', 'Return', 'Date'], axis="columns")
```

2.3.3 Janela Deslizante Mensal

O conjunto de dados é dividido em uma série de janelas de tamanhos iguais, onde cada janela representa um mês (22 dias).Implementação da Janela deslizante mensal:

```
from sklearn.neural_network import MLPRegressor
regressor = MLPRegressor(hidden_layer_sizes=(1), max_iter=100,random_state=1111)
y_pred_window, y_test_window = expanding_window(regressor, X, y, window_size=22)
```

Avaliação do modelo:

```
RMSE =  1.3186453639106654
MAE =  1.0059496298849921
```

A técnica Janela Deslizante Mensal apresenta valores menores de RMSE e MAE, se comparado com o *hold out*.

2.3.4 Janela Deslizante Semanal

O conjunto de dados é dividido em uma série de janelas de tamanhos iguais, onde cada janela representa uma semana (5 dias). Implementação da Janela deslizante mensal:

```
regressor = MLPRegressor(hidden_layer_sizes=(1), max_iter=100,random_state=1111)
y_pred_window, y_test_window = expanding_window(regressor, X, y, window_size=5)
```

Avaliação do modelo:

```
RMSE =  1.4794723216829302
MAE =  1.1516520476149823
```

A técnica Janela Deslizante Semanal apresenta valores maiores de RMSE e MAE, se comparado com o *hold out* e Janela Deslizante Mensal.

2.3.5 Janela Deslizante Diária

O conjunto de dados é dividido em uma série de janelas de tamanhos iguais, onde cada janela representa um dia. Implementação da Janela deslizante mensal:

```
regressor = MLPRegressor(random_state=1111)
y_pred_window, y_test_window = expanding_window(regressor, X, y, window_size=1)
evaluate_regressor(y_test_window, y_pred_window)
```

Avaliação do modelo:

```
RMSE =  1.3246313711992275
MAE =  1.019878099138729
```

A técnica Janela Deslizante Diária apresentou valores menores de RMSE e MAE, se comparado com o *hold out* e Janela Deslizante Semanal.

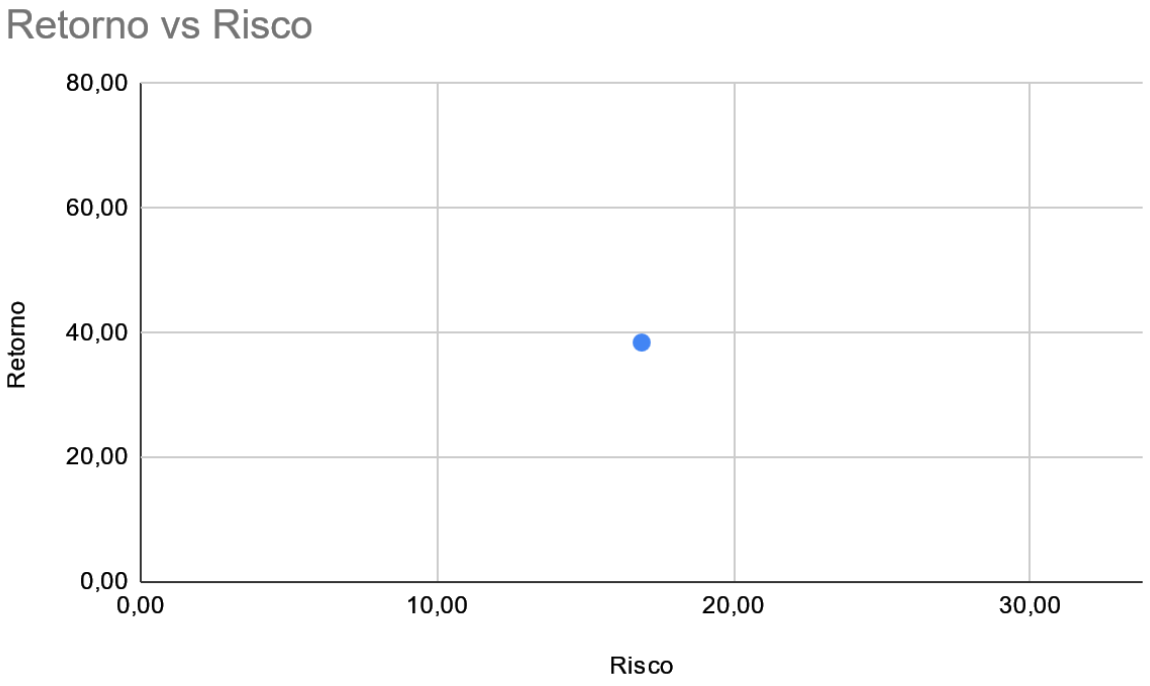
4. Resultados Financeiros

4.1 Classificação

4.1.1 *Hold out*

1. Retorno: 38,45
2. Risco: 16,90
3. Plot do retorno acumulado no período de testes

4. Plot do retorno vs. risco no período de testes

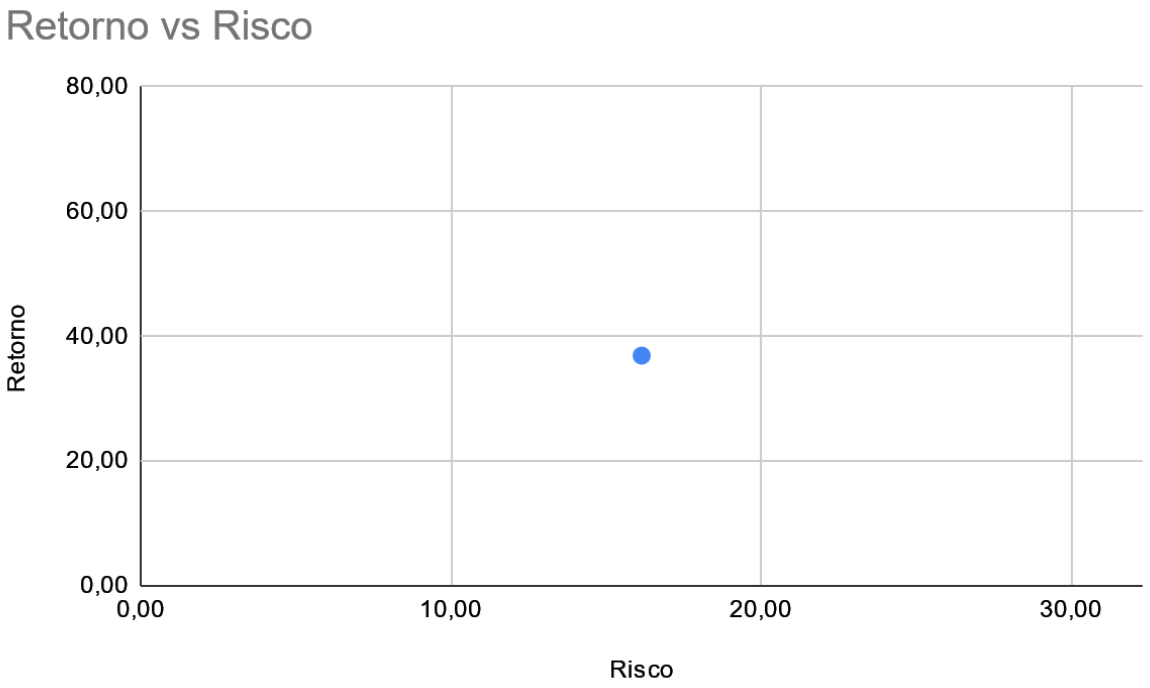


O *hold out* apresentou um pequeno lucro de 17,43, sua máxima foi mais de 60, com um alto risco de 16,90.

4.1.2 Janela Deslizante Mensal

1. Retorno: 36,91
2. Risco: 16,15
3. Plot do retorno acumulado no período de testes

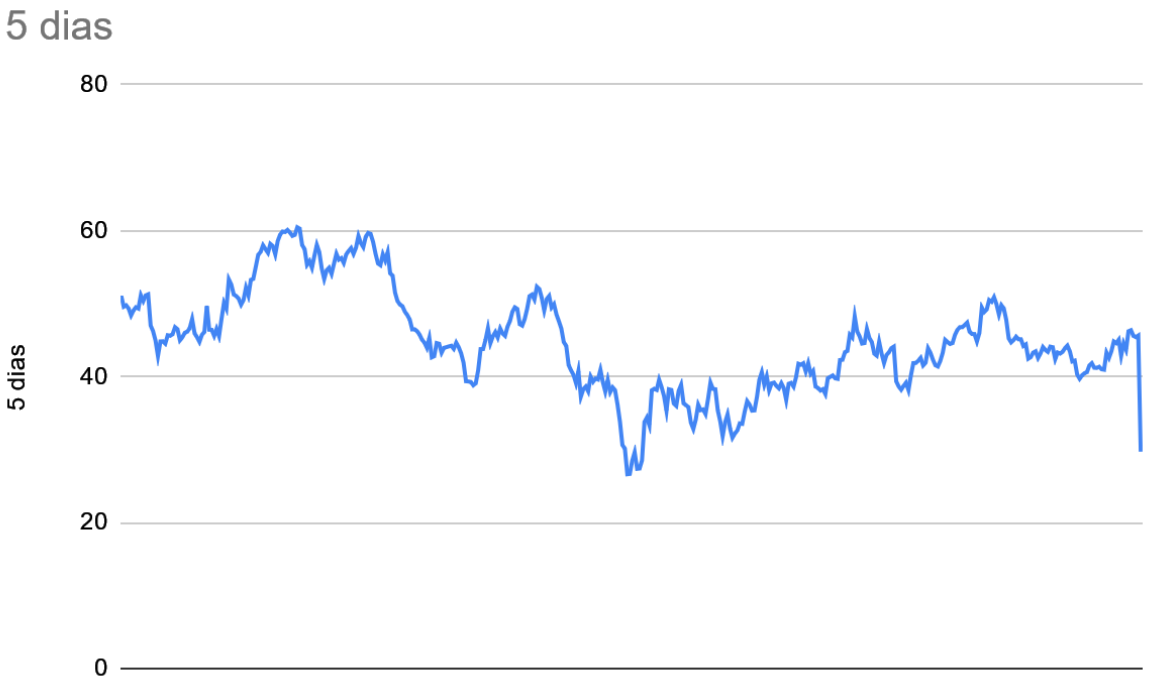
4. Plot do retorno vs. risco no período de testes



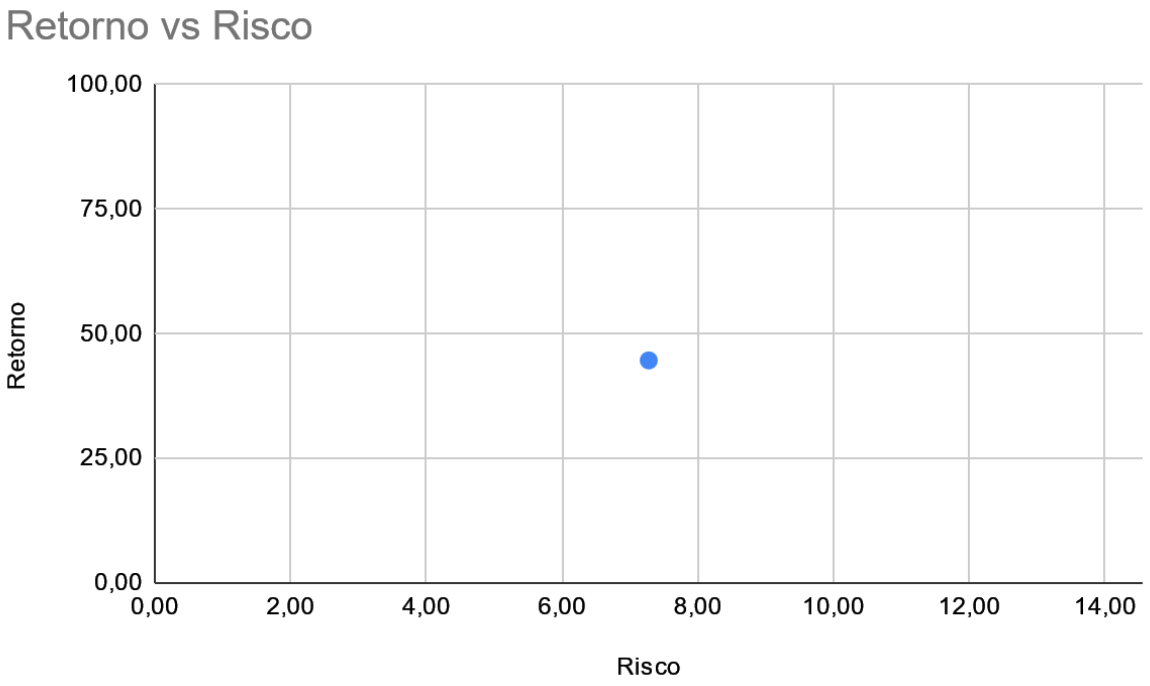
A Janela Deslizante Mensal apresentou um lucro de 23,47 um pouco maior que o *hold out*, sua máxima foi mais de 60, com um alto risco de 16,15.

4.1.3 Janela Deslizante Semanal

1. Retorno: 44,66
2. Risco: 7,28
3. Plot do retorno acumulado no período de testes



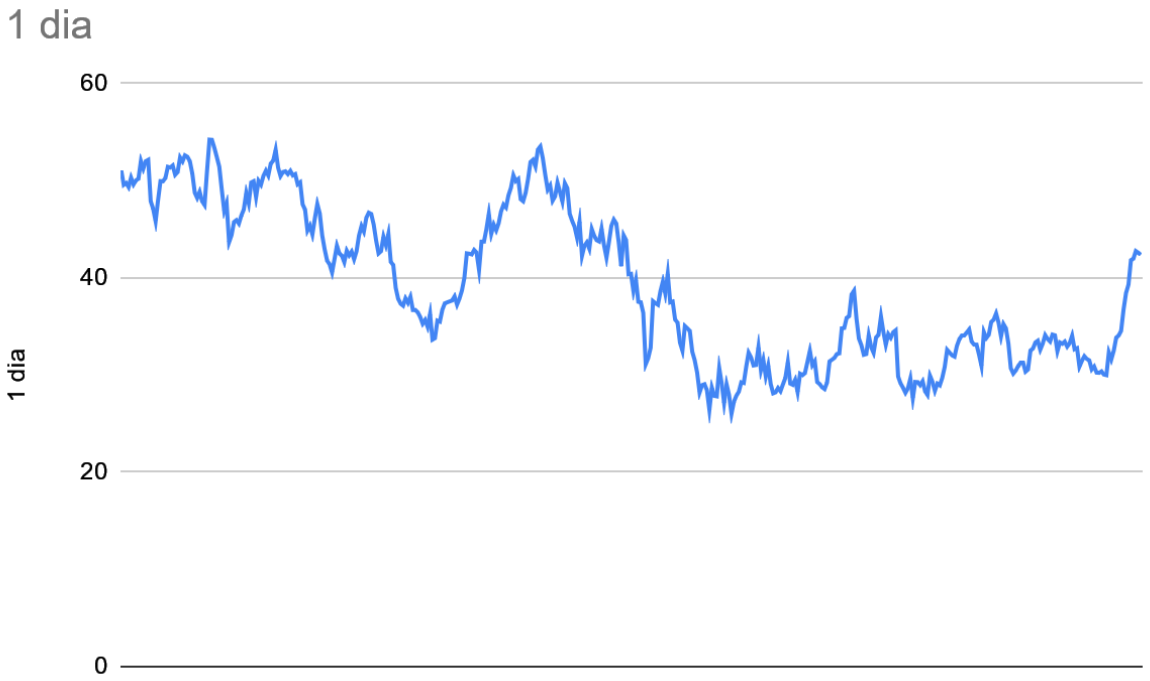
4. Plot do retorno vs. risco no período de testes



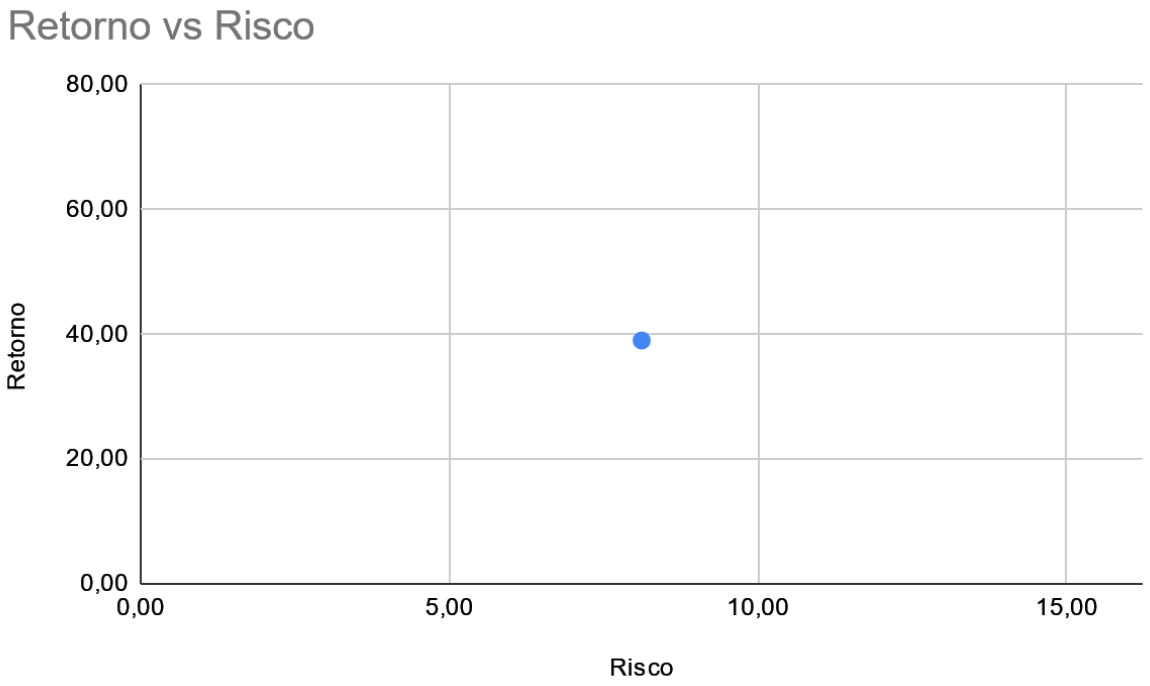
A Janela Deslizante Semanal, se comparada com as outras técnicas, apresentou um lucro alto de 45,57, sua máxima foi um pouco mais de 60, com um risco menor de 7,28.

4.1.4 Janela Deslizante Diária

1. Retorno: 39,00
2. Risco: 8,11
3. Plot do retorno acumulado no período de testes



4. Plot do retorno vs. risco no período de testes

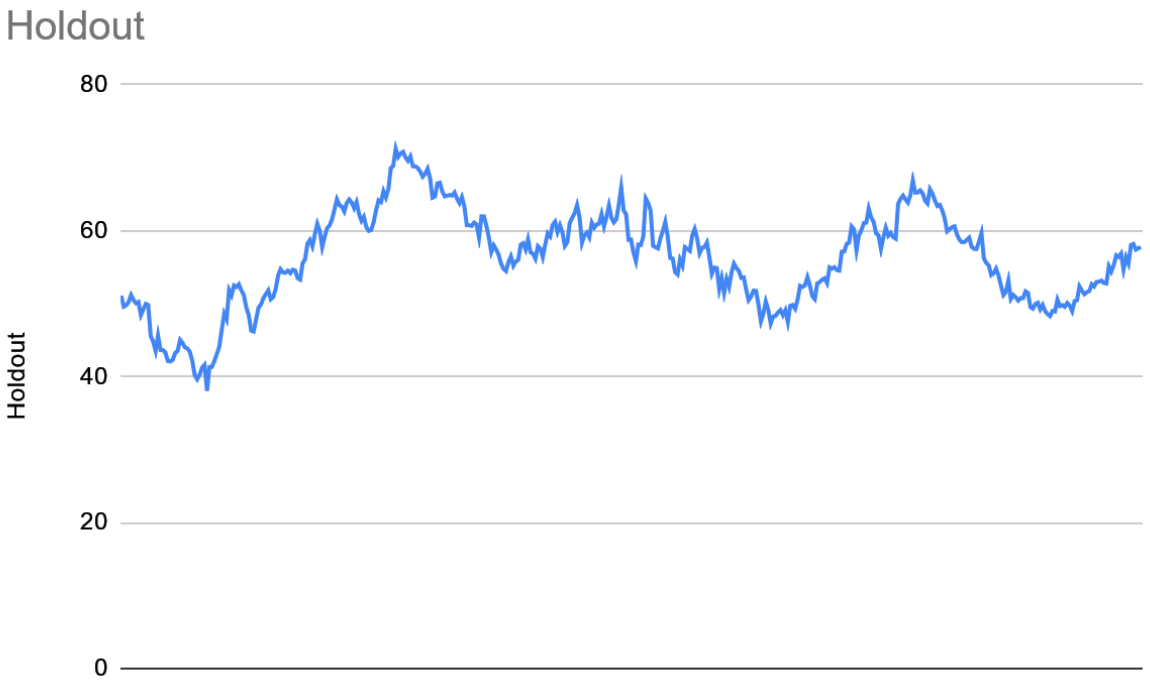


A Janela Deslizante Diária, se comparada com as outras técnicas, apresentou um lucro de 42,31, sua máxima foi mais de 60, semelhante a Janela Deslizante Semanal, com um risco menor de 8,11.

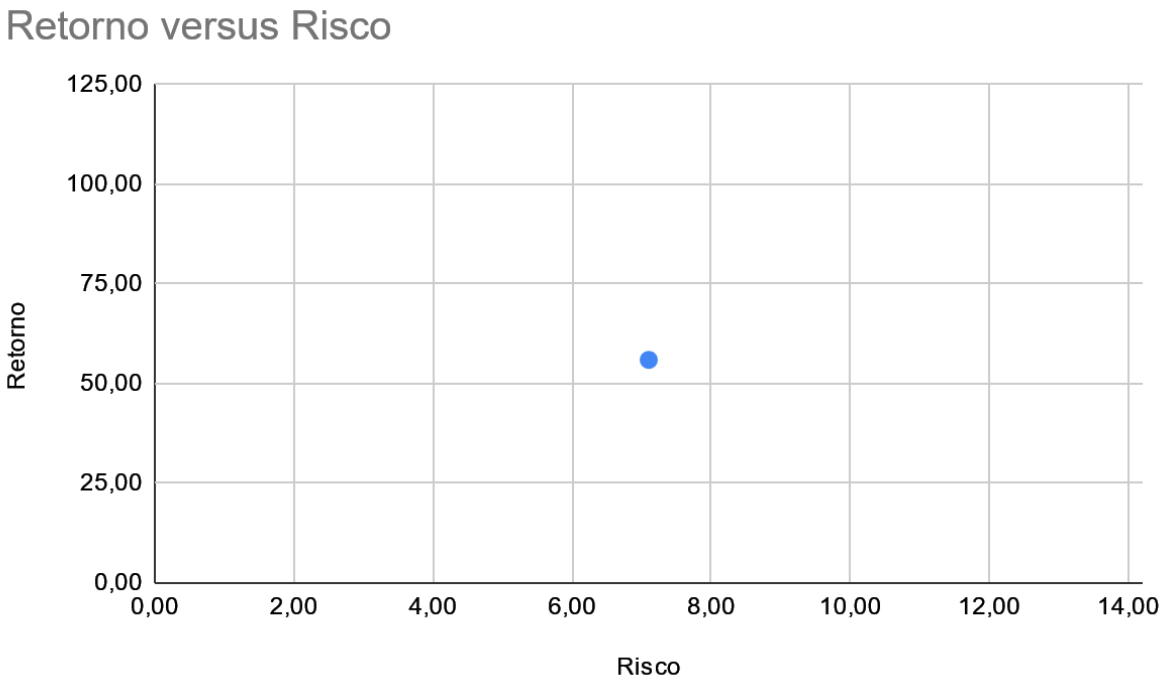
4.2 Regressão

4.2.1 *Hold out*

1. Retorno: 55,99
2. Risco: 7,10
3. Plot do retorno acumulado no período de testes



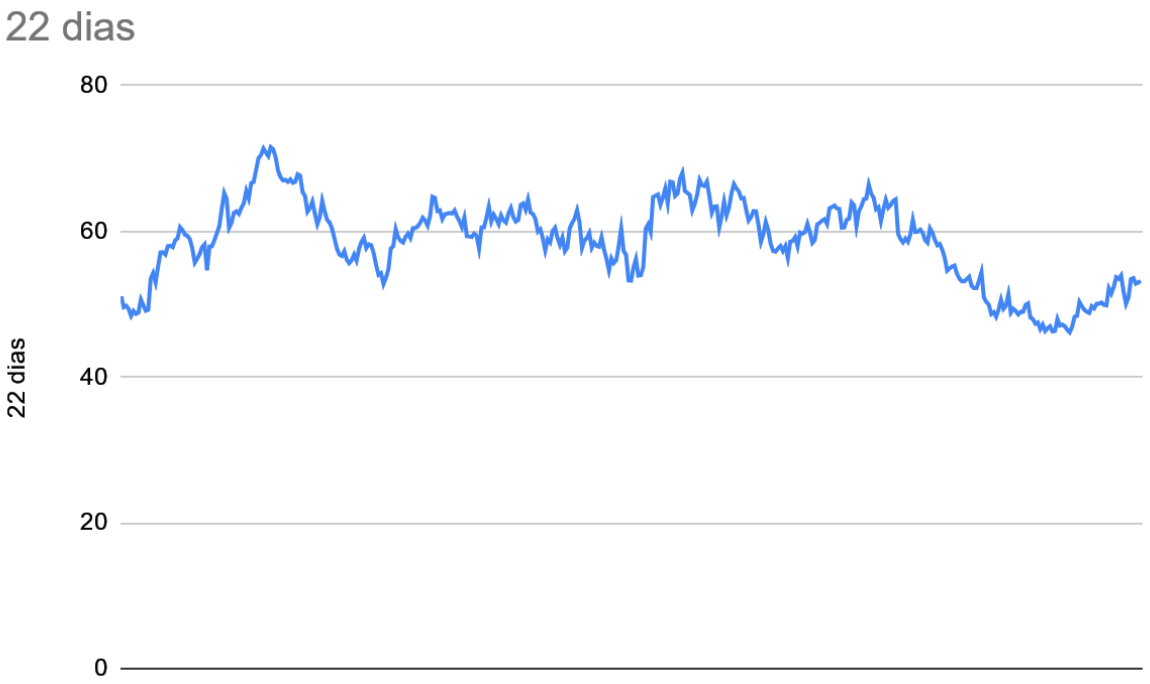
4. Plot do retorno vs. risco no período de testes



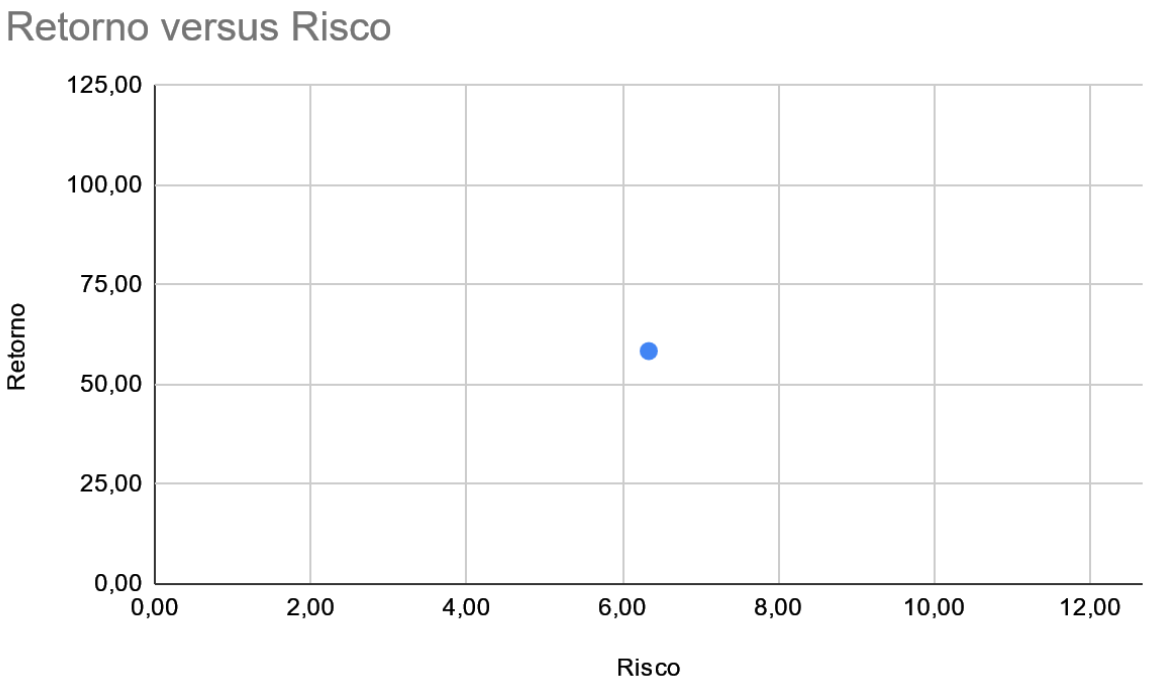
O *hold out* apresentou um lucro de 57,71, o melhor resultado entre as outras técnicas, sua máxima foi mais de 70, com um risco de 7,10.

4.2.1 Janela Deslizante Mensal

1. Retorno: 58,43
2. Risco: 6,34
3. Plot do retorno acumulado no período de testes



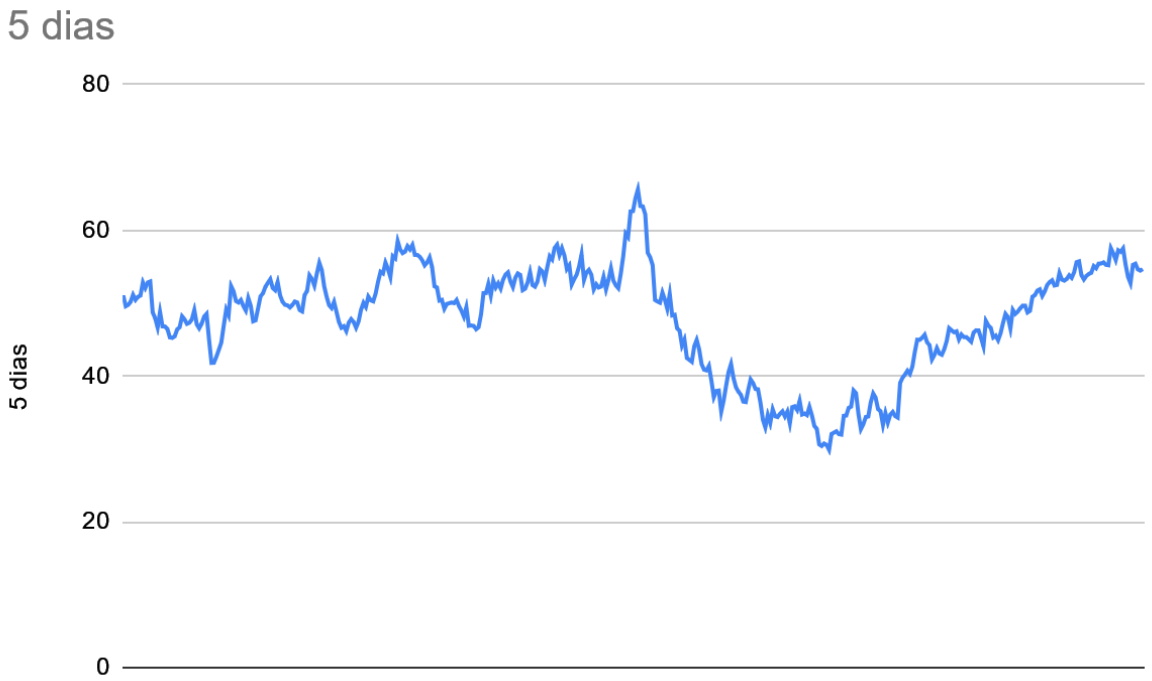
4. Plot do retorno vs. risco no período de testes



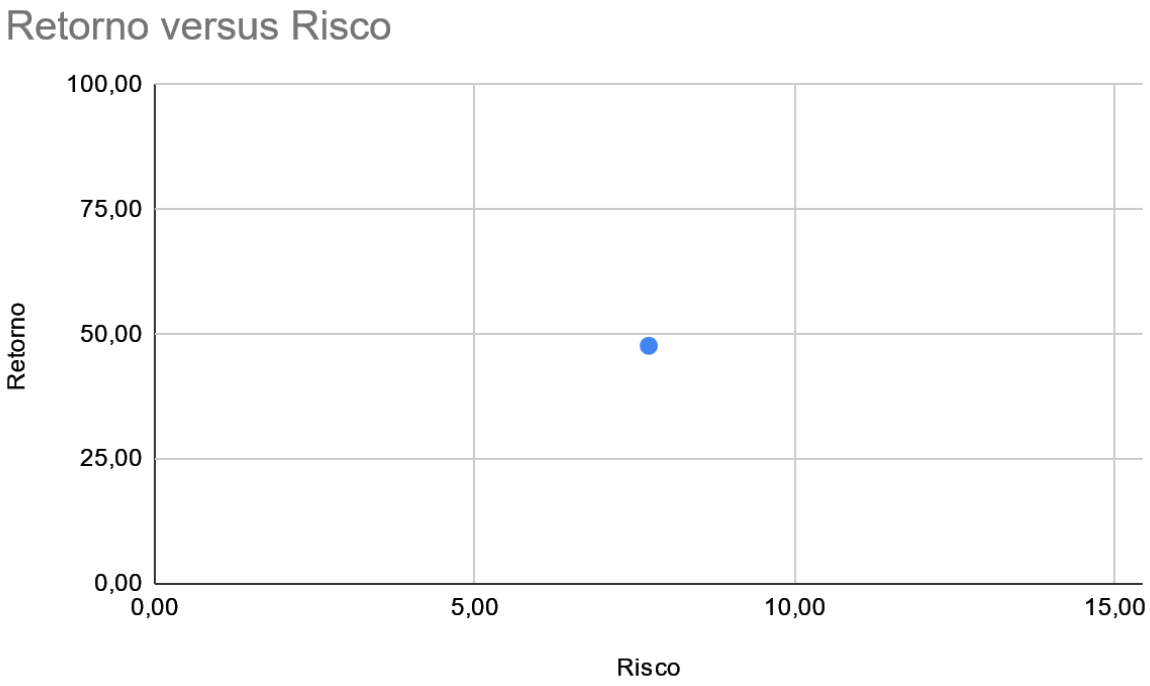
A Janela Deslizante Mensal apresentou um lucro de 53,13 um pouco menor que o *hold out*, sua máxima foi mais de 70, com um risco menor de 6,34.

4.2.3 Janela Deslizante Semanal

1. Retorno: 47,70
2. Risco: 7,71
3. Plot do retorno acumulado no período de testes



4. Plot do retorno vs. risco no período de testes

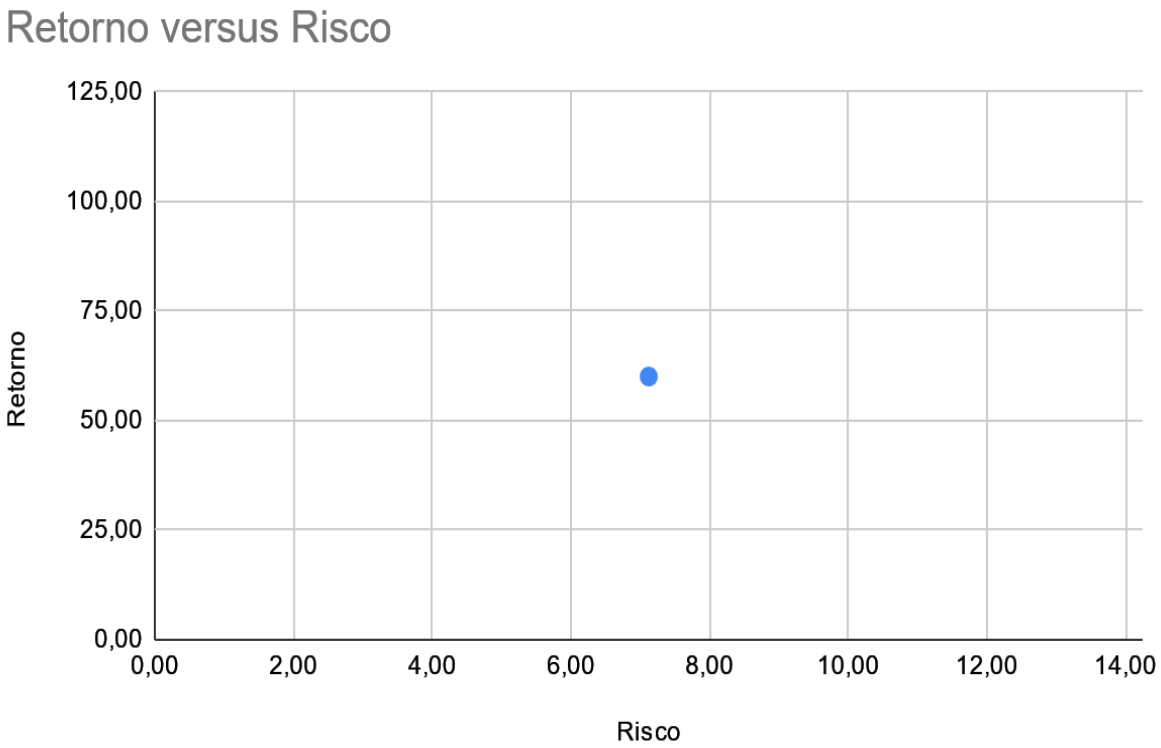


A Janela Deslizante Semanal apresentou um lucro de 54,63 um pouco maior que a Janela Deslizante Mensal e menor que o *hold out*, sua máxima foi mais de 60, com um risco de 7,17.

4.2.4 Janela Deslizante Diária

1. Retorno: 60,07
2. Risco: 7,12
3. Plot do retorno acumulado no período de testes

4. Plot do retorno vs. risco no período de testes



A Janela Deslizante Diária apresentou um lucro de 55,53, semelhante aos resultados anteriores, sua máxima foi mais de quase 80, com um risco de 7,12.