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

TÓPICOS EM SISTEMAS INTELIGENTES

Engenharia de Computação - 8º Período

Computação Financeira 2

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). A etapa de simulação do retorno foi implementada em um projeto Java feito na IDE Netbeans.

A etapa dois, que tem como objetivo, criação de um Ensemble Learning e Stacking, 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

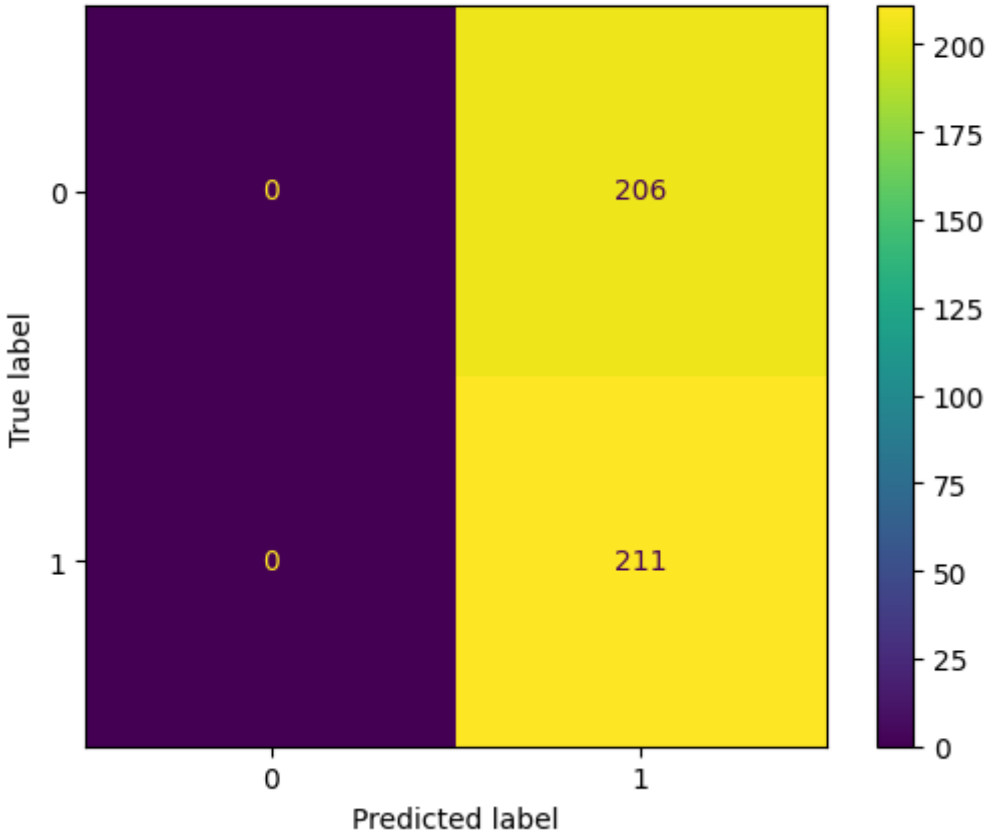
417 rows × 1 columns

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

3.1 Classificação

O modelo de classificação emprega cinco algoritmos distintos: *Random Forest, Multilayer Perceptron, XGBoost, Naive Bayes* e *K-Nearest Neighbors*. Ele foi treinado utilizando um conjunto de dados históricos de preços de ações, utilizando essas informações para aprender a identificar padrões.

3.1.1 Votação

Para combinar os resultados dos modelos treinados previamente, utilizamos a técnica de votação simples dos modelos de aprendizado de máquina. Os dados foram divididos em dois subconjuntos: um de treinamento com informações até 2022 e outro de teste com dados a partir de 2022:

```
# Divide o Conjunto de Dados em subconjuntos de treinamento e teste
y_train_full = df.loc[df['year'] < 2022, 'Output']
X_train_full = 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")
```

Os modelos base são gerados por meio de uma função chamada ***get_models_classification()***:

```
def get_models_classification():
    """Cria lista com modelos base"""
    models = list()
    models.append(('RandomForest', RandomForestClassifier(random_state=22)))
    models.append(('MLP', MLPClassifier(random_state=22)))
    models.append(('XGBoost', XGBClassifier(random_state=22)))
    models.append(('NaiveBayes', GaussianNB()))
    models.append(('KNN', KNeighborsClassifier()))

    return models
```

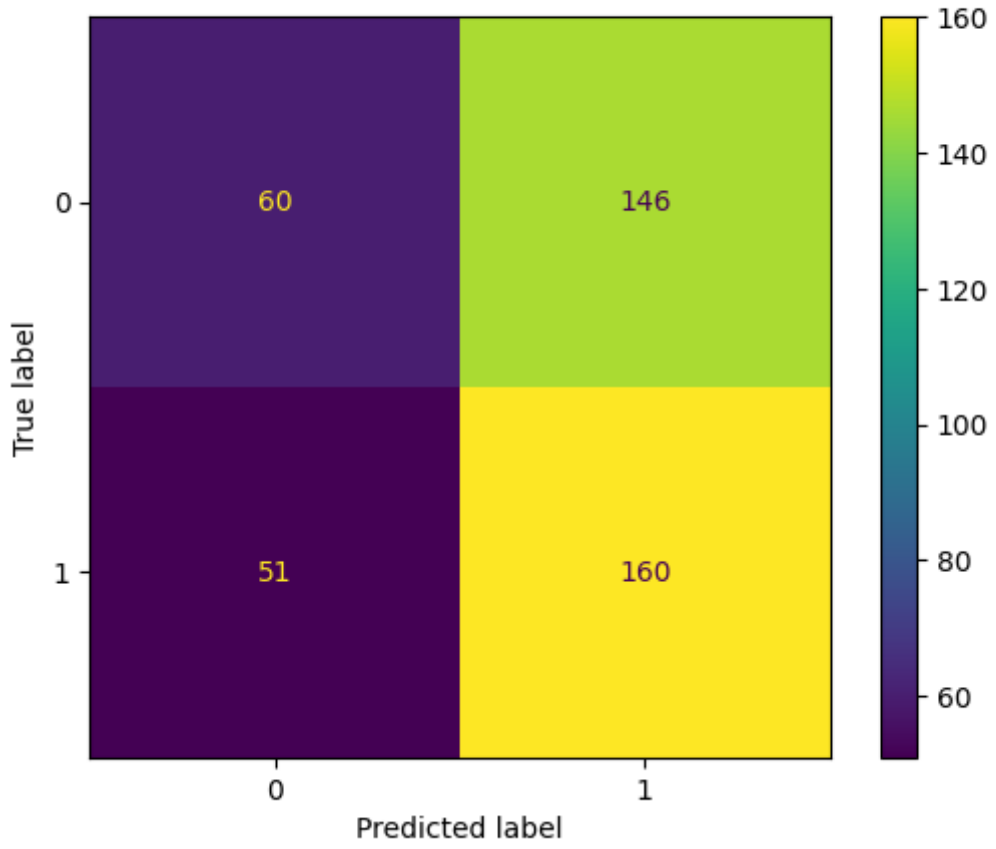
```
# Cria os modelos base
models = get_models_classification()
```

Durante o treinamento do modelo, o ensemble de votação foi criado utilizando o ***VotingClassifier()*** da biblioteca Sklearn. Por padrão, o modo de votação é voto majoritário, enquanto os modelos são fornecidos ao parâmetro estimators

```
# Cria a técnica de Ensemble de Votação
voting = VotingClassifier(estimators=models)

# Aplica Votação para o conjunto de Treinamento
voting.fit(X_train_full, y_train_full)

# Faz previsões no conjunto de Teste
y_pred = voting.predict(X_test)

# Avalia as predições
evaluate_classification(y_test, y_pred)
```

Avaliação do modelo:

```
Accuracy: 0.5275779376498801
Precision: 0.5228758169934641
Recall: 0.7582938388625592
AUC: 0.5247779873924446
```



Em relação à abordagem do Holdout empregada no trabalho prático anterior, que obteve um resultado de AUC de 0,5166, a implementação do ensemble de votação revelou uma leve melhoria.A transição para o ensemble de votação representou uma mudança significativa na estratégia de modelagem. A melhoria no AUC, mesmo que seja modesta, alcançada através do ensemble de votação demonstra a capacidade desse método em aprimorar a capacidade preditiva do modelo.

3.1.2 Média

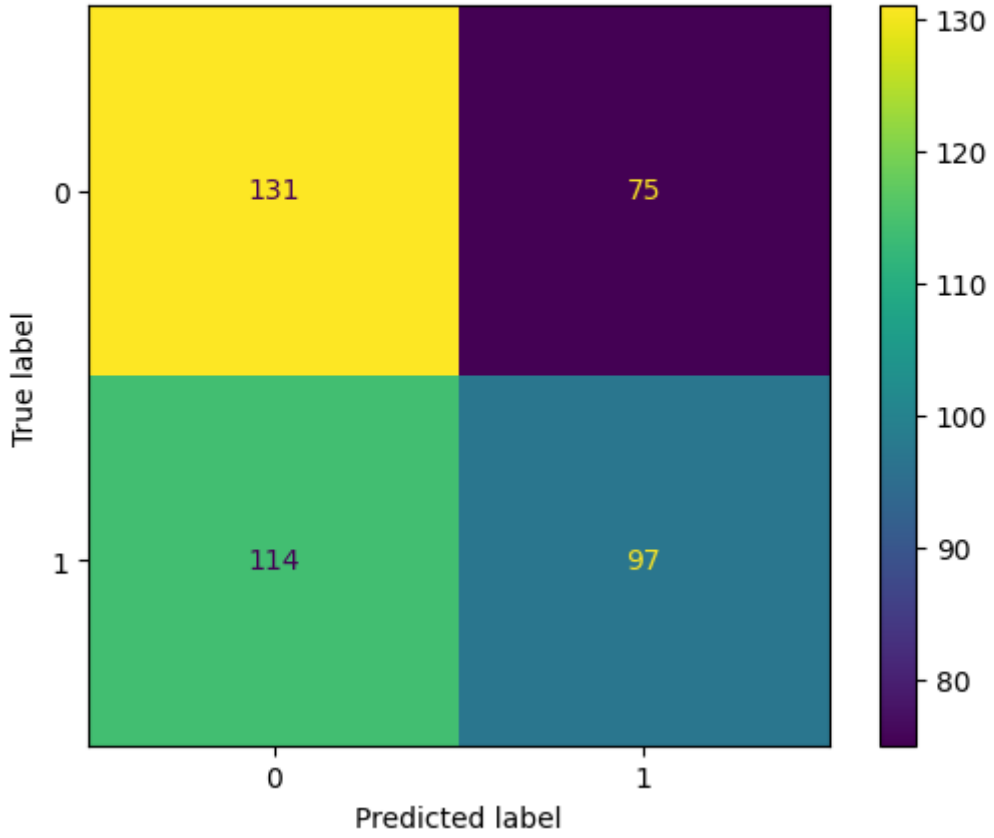
Nesta abordagem, a média das saídas dos modelos de aprendizado de máquina é calculada, aplicando um limiar arbitrário para gerar a saída binária. O conjunto de dados foi dividido de forma semelhante à técnica de votação.

Durante o treinamento do modelo, o ensemble por média foi implementado utilizando o ***VotingClassifier()*** da biblioteca Sklearn, com os modelos sendo fornecidos ao parâmetro estimators. O modo de votação escolhido foi *'soft'*, pois essa configuração utiliza a média para prever as saídas.

```
# Cria a técnica de Ensemble por Média
average = VotingClassifier(estimators=models, voting='soft')

# Aplica Média para o conjunto de Treinamento
average.fit(X_train_full, y_train_full)

# Faz previsões no conjunto de Teste
y_pred = average.predict(X_test)

# Avalia as predições
evaluate_classification(y_test, y_pred)
```

Avaliação do modelo:

```
Accuracy: 0.5467625899280576
Precision: 0.563953488372093
Recall: 0.4597156398104265
AUC: 0.5478189849537569
```



Comparando diferentes métodos de ensemble para a avaliação do desempenho, observamos que o método de votação inicialmente apresentou um resultado de AUC de 0,5247. No entanto, ao empregar a técnica de ensemble por média, notou-se uma melhoria sutil, mas significativa, com um AUC aprimorado de 0,54781.

Esta discreta, porém notável, melhora na métrica de avaliação destaca a eficácia do ensemble por média na consolidação das previsões dos diferentes modelos. Esse aumento no AUC, embora possa parecer modesto, muitas vezes representa melhorias substanciais na capacidade do modelo de generalizar e distinguir entre classes, especialmente em contextos onde até pequenas variações podem ter um impacto significativo nas decisões baseadas nos resultados preditivos.

Assim, a transição para o ensemble por média revelou-se uma estratégia eficaz para otimizar a performance do modelo, evidenciando a importância de considerar e explorar diferentes métodos de ensemble na busca por melhorias incrementais no desempenho preditivo.

3.1.3 Média Ponderada

Neste método, calculamos a média ponderada das saídas dos modelos de aprendizado de máquina, aplicando um limiar arbitrário para gerar a saída binária. Os pesos são determinados usando um conjunto de validação, o qual foi obtido após dividirmos os dados em três subconjuntos: treinamento com informações até 2021, validação de 2021 até 2022 e teste com dados a partir de 2022:

```
y_train_full = df.loc[df['year'] < 2022, 'Output']
X_train_full = 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")

# Divide o Conjunto de Dados em subconjuntos de treinamento e validação
y_train = df.loc[df['year'] < 2021, 'Output']
X_train = df.loc[df['year']  < 2021].drop(['Output', 'Return', 'Date'], axis="columns")

y_val = df.loc[(df['year']  >= 2021) & (df['year']  < 2022), 'Output']
X_val = df.loc[(df['year']  >= 2021) & (df['year']  < 2022)].drop(['Output', 'Return', 'Date'], axis="columns")
```

Os pesos foram calculados por meio da função criada denominada ***evaluate_models()***.

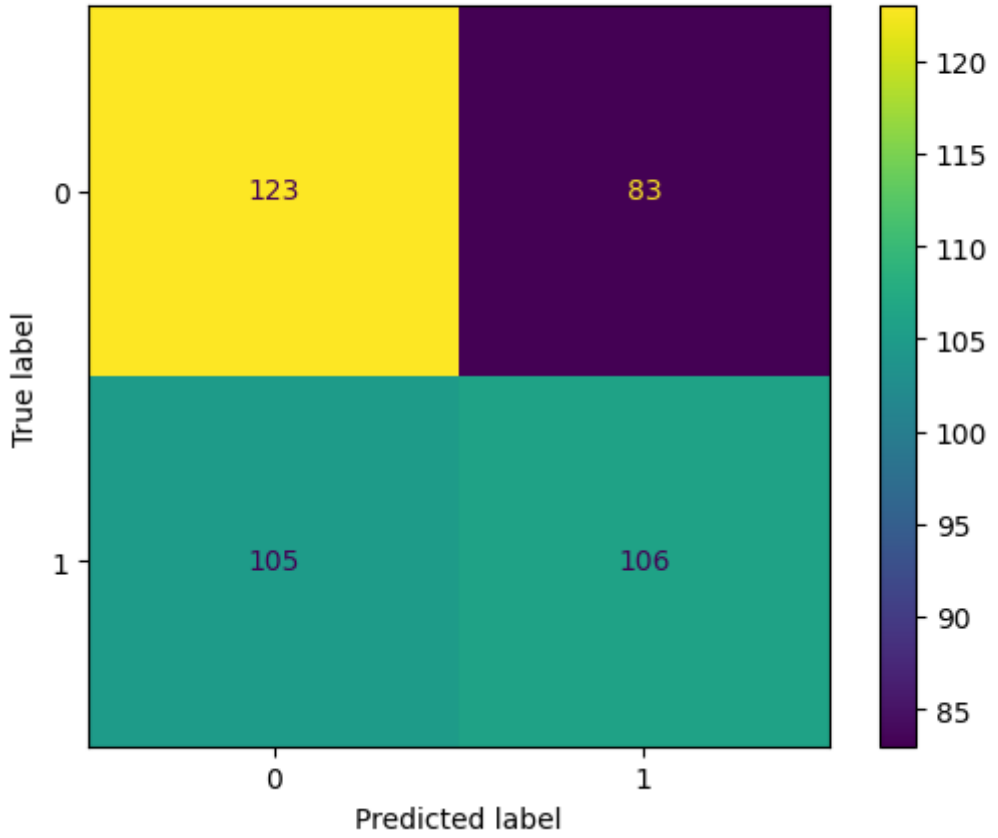
```
def evaluate_models(models, X_train, X_val, y_train, y_val):
    """Treina e avalia cada um dos modelos base e retorna suas pontuações AUC e previsões dos modelos"""

    scores = list()
    predictions = []

    # Treina e avalia os modelos
    for name, model in models:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_val)

        auc = roc_auc_score(y_val, y_pred)
        scores.append(auc)
        predictions.append(y_pred)

        print(f"{name}: {auc}\n")

    return scores, predictions
```

Durante o treinamento do modelo, o ensemble por média foi implementado utilizando o ***VotingClassifier()*** da biblioteca Sklearn, com os modelos sendo fornecidos ao parâmetro *estimators*. Optamos pelo modo de votação *'soft'*, o qual utiliza a média para prever as saídas, e os pesos foram atribuídos ao parâmetro *weights*.

```
# Calcula as pontuações AUC e previsões dos modelos
scores, pred = evaluate_models(models, X_train, X_val, y_train, y_val)

# Cria a técnica de Ensemble por Média Ponderada
weighted_average = VotingClassifier(estimators=models, voting='soft', weights=scores)

# Aplica Média Ponderada para o conjunto de Treinamento
weighted_average.fit(X_train_full, y_train_full)

# Faz previsões no conjunto de Teste
y_pred = weighted_average.predict(X_test)

# Avalia as predições
evaluate_classification(y_test, y_pred)
```

Avaliação do modelo:

```
Accuracy: 0.5491606714628298
Precision: 0.5608465608465608
Recall: 0.5023696682464455
AUC: 0.5497285234436111
```



3.1.4 Stacking

Foi empregado um modelo de aprendizado supervisionado para combinar as saídas dos modelos mencionados anteriormente. Utilizamos um classificador *Multilayer Perceptron* (MLP) nesta etapa para agregar as saídas dos modelos. Para criar o modelo de stacking, empregamos dados do período entre 2021 e 2022 para gerar as saídas dos modelos individuais. Posteriormente, unificamos esses dados com a saída esperada e aplicamos um classificador MLP.

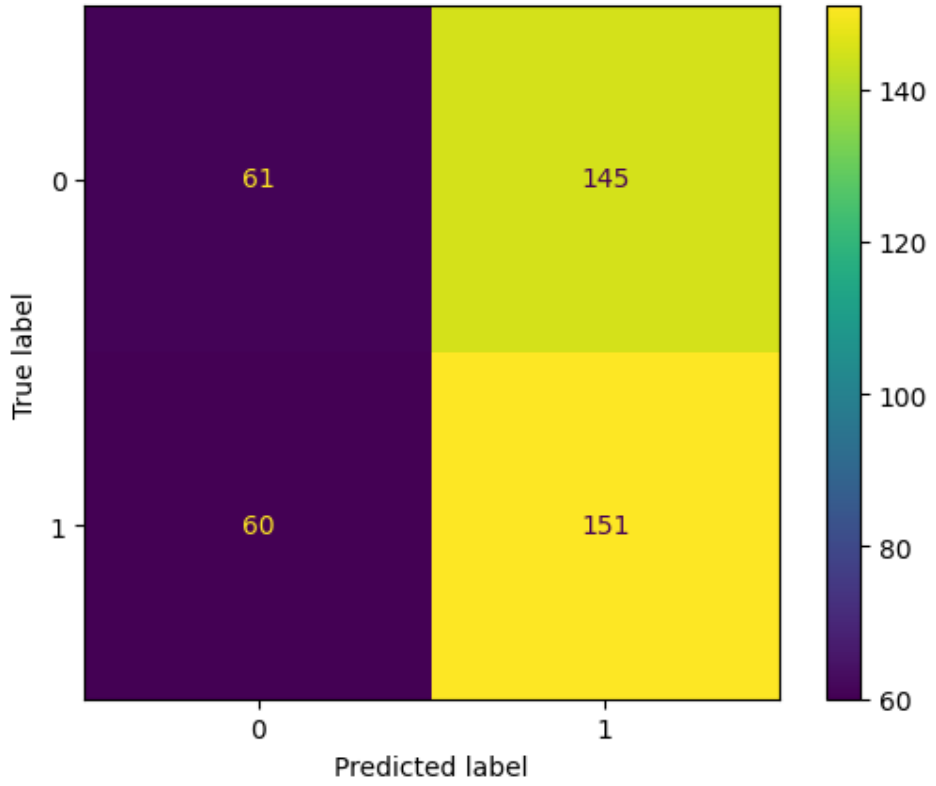
```python
# Calcula as pontuações AUC e previsões dos modelos com o conjunto de validação
scores, pred = evaluate_models(models, X_train, X_val, y_train, y_val)

# Gera DataFrame com as saídas dos modelos individuais
df_stacking = pd.DataFrame(data=np.transpose(pred), columns=['RandomForest', 'MLP', 'XGBoost', 'NaiveBayes', 'KNN'])
df_stacking['Output'] = pd.DataFrame(y_val).reset_index().drop(['index'], axis=1)

y_train_stacking = df_stacking['Output']
X_train_stacking = df_stacking.drop(['Output'], axis="columns")

# Cria a técnica de Stacking
stacking = MLPClassifier(random_state=22)

# Aplica Stacking para o conjunto de validação
stacking.fit(X_train_stacking, y_train_stacking)
```

Uma vez que o modelo foi treinado, o aplicamos nos dados do conjunto de testes para avaliação.

```python
# Calcula as pontuações AUC e previsões dos modelos com o conjunto de teste
scores, pred = evaluate_models(models, X_train_full, X_test, y_train_full, y_test)

# Faz previsões no conjunto de Teste
X_test_stacking = pd.DataFrame(data=np.transpose(pred), columns=['RandomForest', 'MLP', 'XGBoost', 'NaiveBayes', 'KNN'])
y_pred_stacking = stacking.predict(X_test_stacking)

# Avalia as predições
evaluate_classification(y_test, y_pred_stacking)
```

Avaliação do modelo:

```
Accuracy: 0.5083932853717026
Precision: 0.5101351351351351
Recall: 0.7156398104265402
AUC: 0.5058781576404545
```



Comparando os resultados entre o método de média, que inicialmente registrou um AUC de 0,54781, e o ensemble por média ponderada, que obteve um AUC um pouco superior, atingindo 0,54972, percebemos uma sutil melhoria. Embora a diferença entre os valores de AUC possa parecer mínima à primeira vista, essa ligeira melhoria é um reflexo da capacidade do ensemble por média ponderada em otimizar ainda mais as previsões agregadas.

3.2 Regressão

O modelo de regressão é utilizado para prever um valor numérico, como o retorno de uma ação.

3.2.1 Média

Seguiu-se os mesmo passos do modelo de classificação, as únicas diferenças foram que para a tarefa de regressão, alguns ajustes tiveram que ser feitos. O conjunto de dados foi dividido em dois subconjuntos:

```
# Divide o Conjunto de Dados em subconjuntos de treinamento e teste
y_test_aux = df.loc[df['year'] >= 2022, ['Date', 'Return']].reset_index()

y_train_full = df.loc[df['year'] < 2022, 'Return']
X_train_full = 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")
```

Os algoritmos de classificação foram convertidos para os de regressão, com exceção do *Naive Bayes*, que foi deixado de fora pois não funciona em tarefas de regressão. Os modelos base são gerados por meio de uma função chamada ***get_models_regression()***:

```
def get_models_regression():
  """Cria lista com modelos base"""
  models = list()
  models.append(('RandomForest', RandomForestRegressor(random_state=22)))
  models.append(('MLP', MLPRegressor(random_state=22)))
  models.append(('XGBoost', XGBRegressor(random_state=22)))
  #models.append(('NaiveBayes', GaussianNB()))
  models.append(('KNN', KNeighborsRegressor()))

  return models
```

```
# Cria os modelos base
models = get_models_regression()
```

Treinamento do modelo:

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

```
# Cria a técnica de Ensemble por Média
average = VotingRegressor(estimators=models)

# Aplica Média para o conjunto de Treinamento
average.fit(X_train_full, y_train_full)

# Faz previsões no conjunto de Teste
y_pred = average.predict(X_test)

# Avalia as predições
evaluate_regressor(y_test, y_pred)
```

Avaliação do modelo:

```
RMSE =  1.389731674044984
MAE =  1.0766153529600835
```

A técnica de Média exibe resultados de RMSE (Root Mean Square Error) e MAE (Mean Absolute Error) que se equiparam de forma comparativa com as outras estratégias exploradas no trabalho prático anterior, como o *holdout*.

Ao analisarmos a performance da técnica de Média em relação a outras abordagens utilizadas previamente, é perceptível que os valores de RMSE e MAE se mantêm em um patamar similar. Essa constatação sugere que, em termos de precisão e acurácia preditiva, a técnica de Média não apresentou uma diferença substancial em relação às abordagens anteriores investigadas no contexto do estudo.

3.2.2 Média Ponderada

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:

```
# Divide o Conjunto de Dados em subconjuntos de treinamento e validação
y_train = df.loc[df['year'] < 2021, 'Return']
X_train = df.loc[df['year']  < 2021].drop(['Output', 'Return', 'Date'], axis="columns")

y_val = df.loc[(df['year']  >= 2021) & (df['year']  < 2022), 'Return']
X_val = df.loc[(df['year']  >= 2021) & (df['year']  < 2022)].drop(['Output', 'Return', 'Date'], axis="columns")
```

Treinamento do Modelo:

```
# Calcula as pontuações AUC e previsões dos modelos
scores, pred = evaluate_models_regression(models, X_train, X_val, y_train, y_val)

# Cria a técnica de Ensemble por Média Ponderada
weighted_average = VotingRegressor(estimators=models, weights=scores)

# Aplica Média Ponderada para o conjunto de Treinamento
weighted_average.fit(X_train_full, y_train_full)

# Faz previsões no conjunto de Teste
y_pred = weighted_average.predict(X_test)

# Avalia as predições
evaluate_regressor(y_test, y_pred)
```

Avaliação do modelo:

```
RMSE =  1.430355470019368
MAE =  1.1026778319941584
```

A abordagem da Média Ponderada oferece uma variação ao ponderar as previsões dos modelos com base em diferentes critérios. Embora essa técnica ainda produza resultados competitivos, se observou valores ligeiramente superiores de RMSE e MAE em comparação com a Média. A inclusão de pesos ajustáveis para cada modelo na composição da previsão agregada pode introduzir uma camada adicional de complexidade e, em alguns casos, pode resultar em um desempenho levemente inferior em comparação com a simplicidade direta da técnica de Média.

3.2.3 Stacking

O modelo de regressão adotou uma abordagem semelhante àquela utilizada no modelo de aprendizado supervisionado, visando combinar as saídas dos modelos anteriores no contexto da modelagem de classificação.

```python
# Calcula as pontuações AUC e previsões dos modelos com o conjunto de validação
scores, pred = evaluate_models_regression(models, X_train, X_val, y_train, y_val)

# Gera DataFrame com as saídas dos modelos individuais
df_stacking = pd.DataFrame(data=np.transpose(pred), columns=['RandomForest', 'MLP', 'XGBoost', 'KNN'])
df_stacking['Output'] = pd.DataFrame(y_val).reset_index().drop(['index'], axis=1)

y_train_stacking = df_stacking['Output']
X_train_stacking = df_stacking.drop(['Output'], axis="columns")

# Cria a técnica de Stacking
stacking = MLPRegressor(random_state=22)

# Aplica Stacking para o conjunto de validação
stacking.fit(X_train_stacking, y_train_stacking)
```

```python
# Calcula as pontuações AUC e previsões dos modelos com o conjunto de teste
scores, pred = evaluate_models_regression(models, X_train_full, X_test, y_train_full, y_test)

# Faz previsões no conjunto de Teste
X_test_stacking = pd.DataFrame(data=np.transpose(pred), columns=['RandomForest', 'MLP', 'XGBoost', 'KNN'])
y_pred_stacking = stacking.predict(X_test_stacking)

# Avalia as predições
evaluate_regressor(y_test, y_pred_stacking)
```

Avaliação do modelo:

```
RMSE =  1.3100101229938386
MAE =  1.0091619368937905
```

A técnica de Média demonstra os menores valores de RMSE e MAE em comparação com outras abordagens.

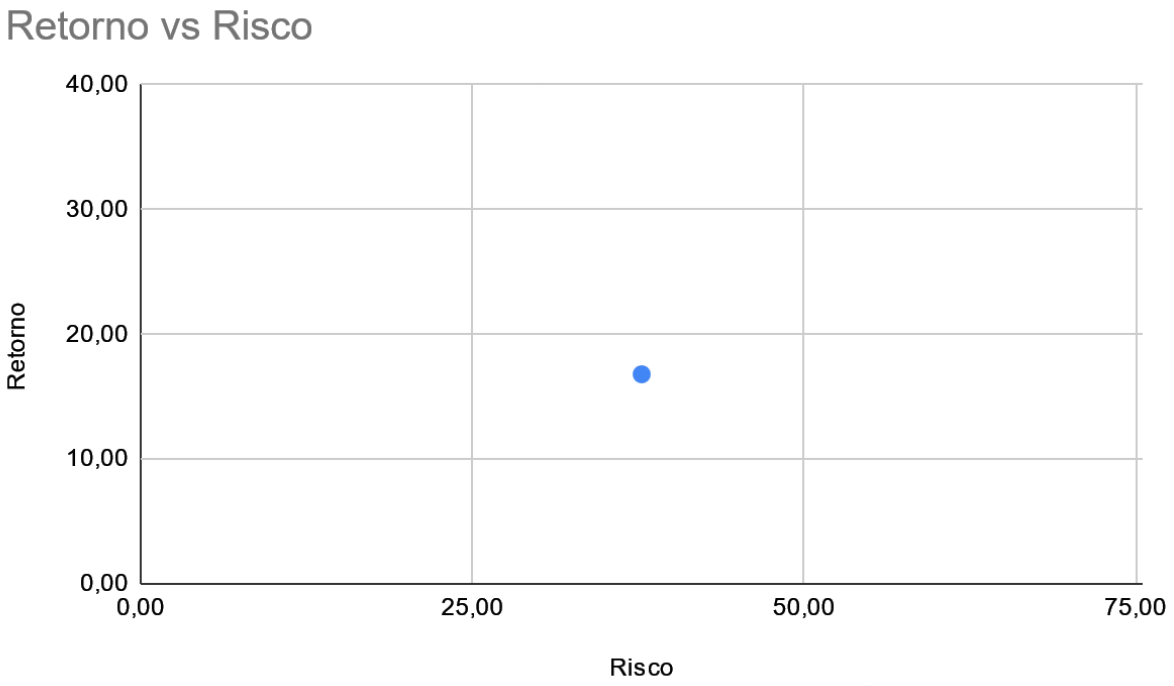
4. Resultados Financeiros

4.1 Classificação

4.1.1 Votação

1. Retorno: 37,76
2. Risco: 16,79
3. Plot do retorno acumulado no período de testes

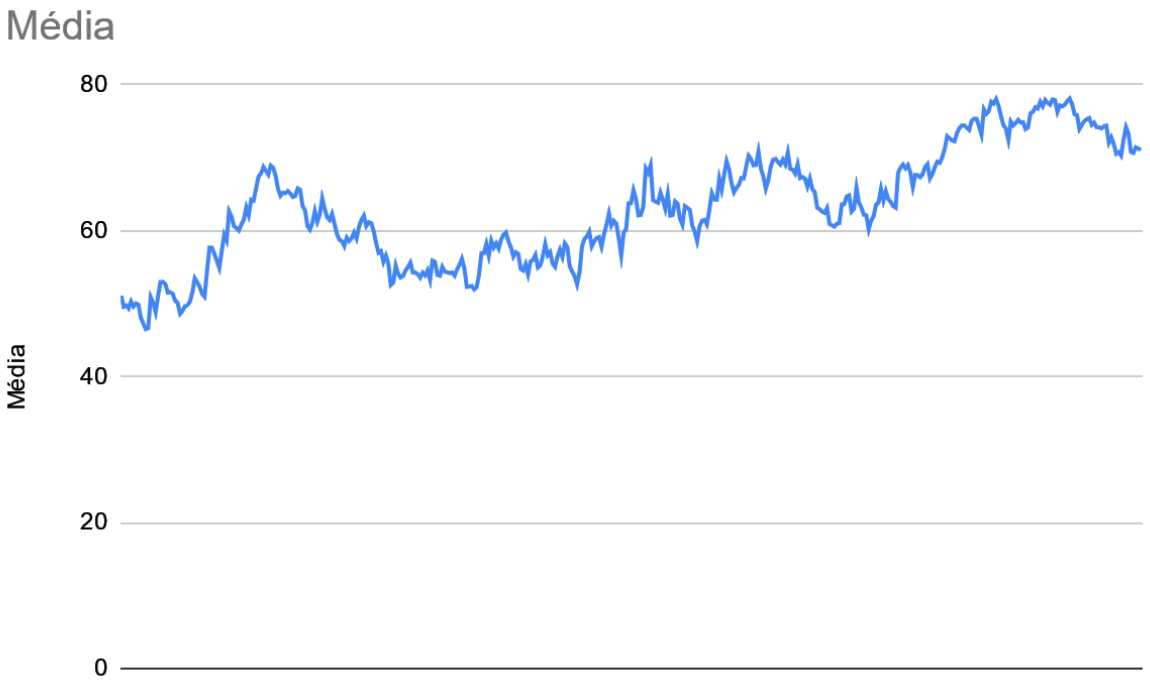
4. Plot do retorno vs. risco no período de testes



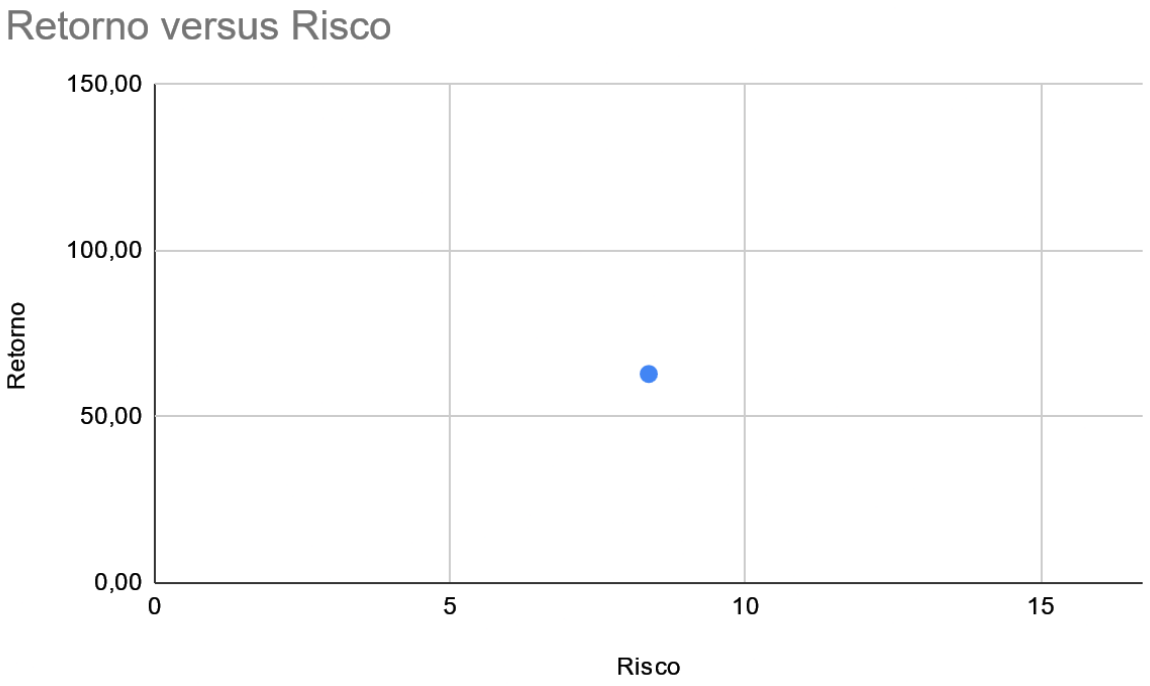
A Votação apresentou um lucro de 54,63, sua máxima foi de mais de 60, com um risco de 16,79.

4.1.2 Média

1. Retorno: 62,94
2. Risco: 8,36
3. Plot do retorno acumulado no período de testes



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

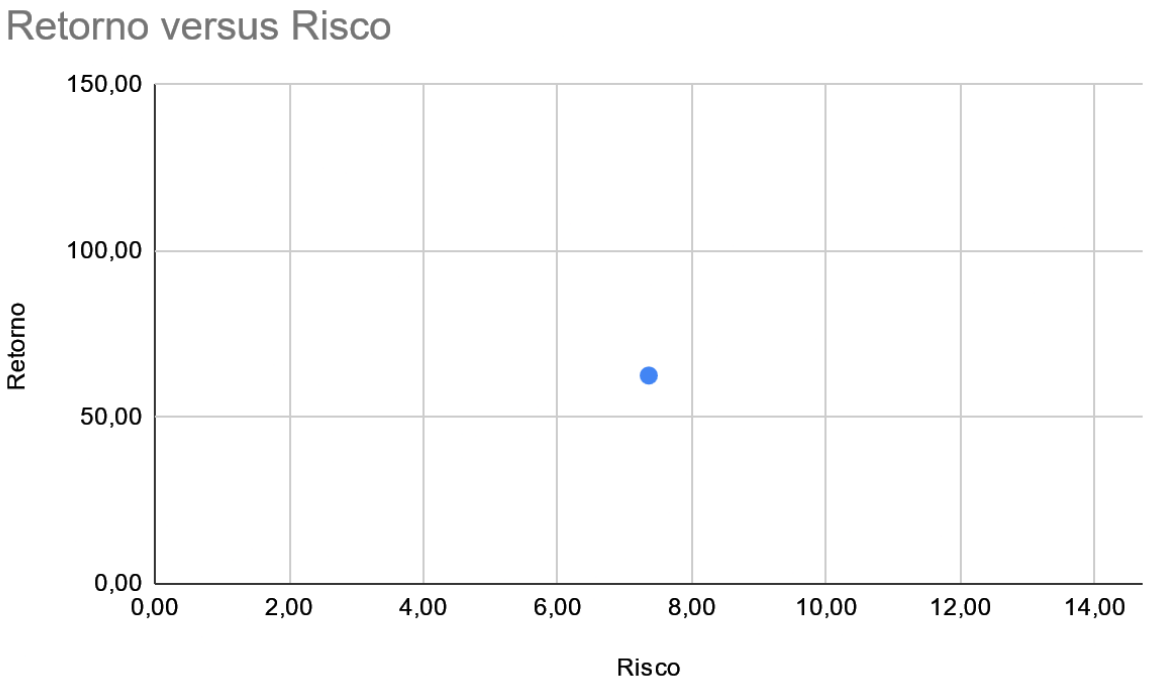


A Média apresentou um lucro de 62,94 um pouco maior que a Média Ponderada, sua máxima foi de quase de 80, com um risco de 8,36.

4.1.3 Média Ponderada

1. Retorno: 62,58
2. Risco: 7,35
3. Plot do retorno acumulado no período de testes

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

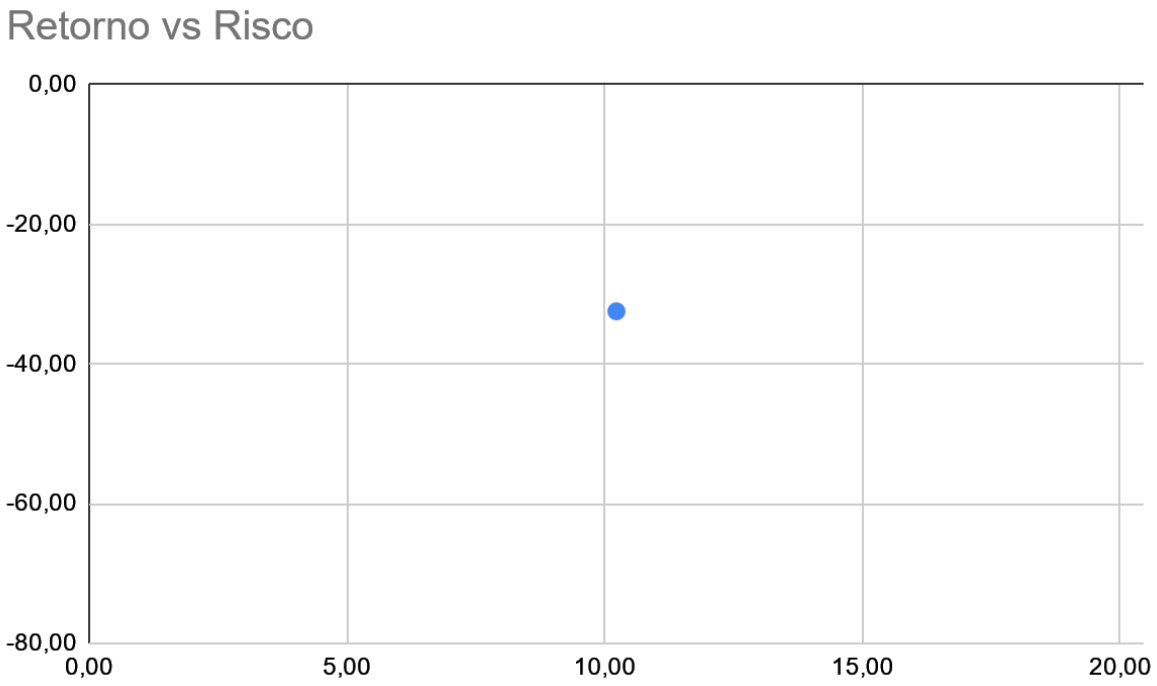


A Média Ponderada apresentou um lucro de 62,58, sua máxima foi de quase de 80, com um risco de 7,35.

4.1.4 Stacking

1. Retorno: -32,46
2. Risco: 10,23
3. Plot do retorno acumulado no período de testes

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

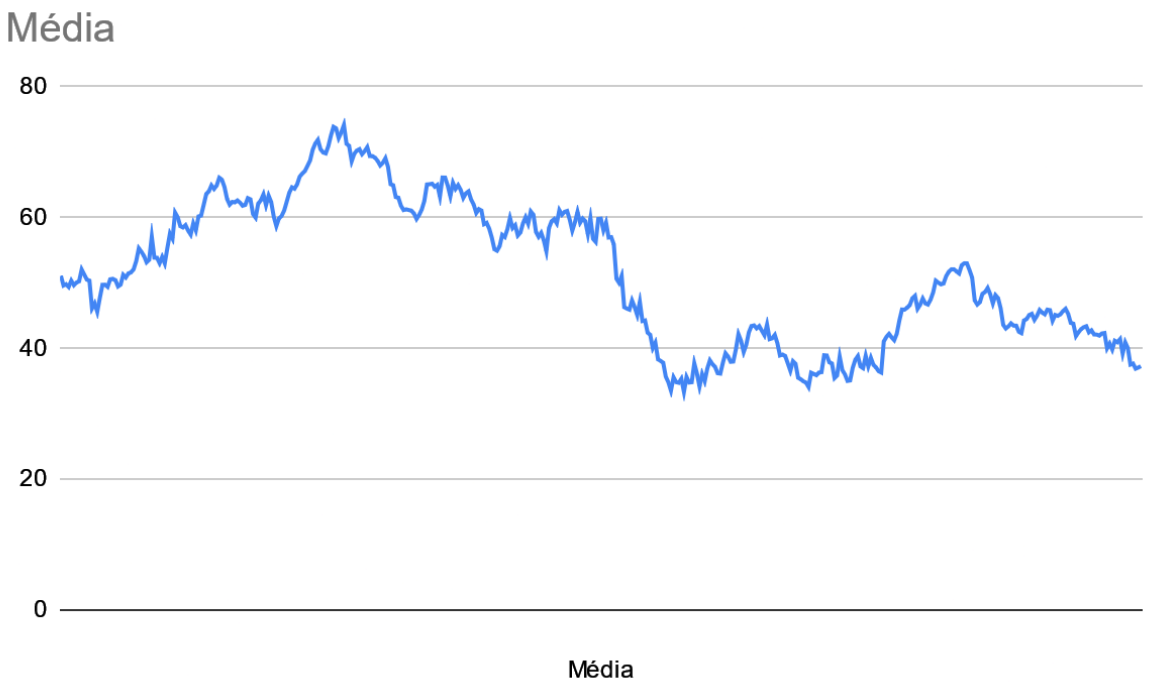


O Stacking apresentou um péssimo resultado com um prejuízo de -32,46, sua máxima não chegou a nem ser positiva, com o segundo maior risco dos modelos de classificação de 10,23.

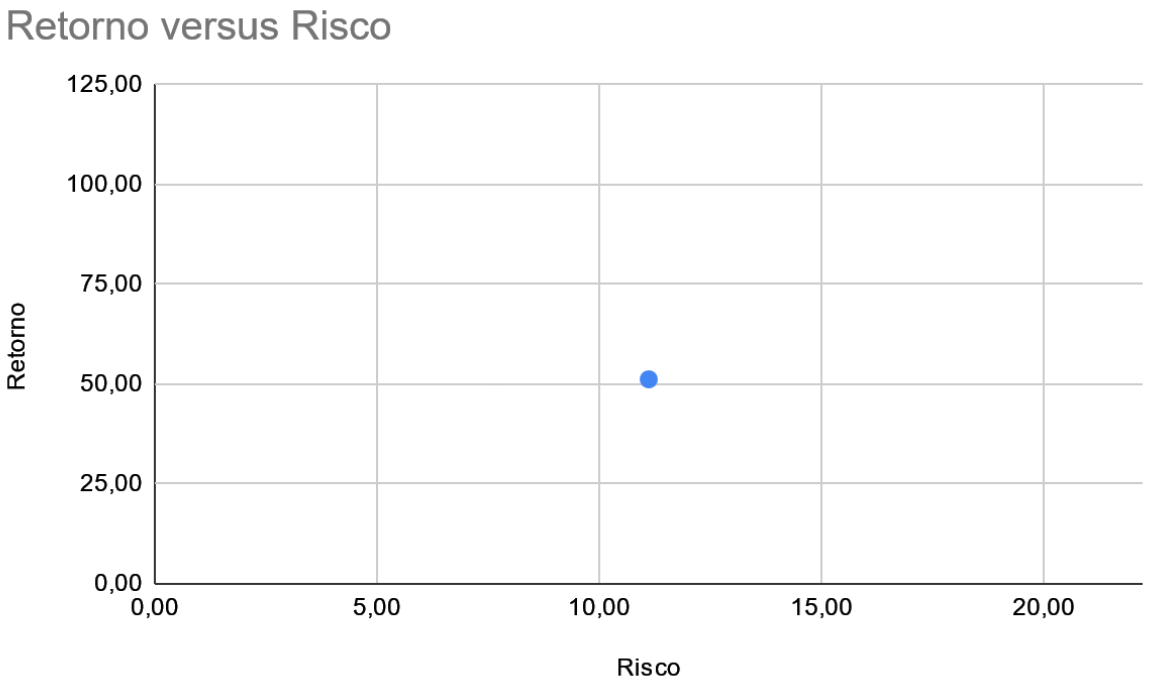
4.2 Regressão

4.2.1 Média

1. Retorno: 51,22
2. Risco: 11,11
3. Plot do retorno acumulado no período de testes



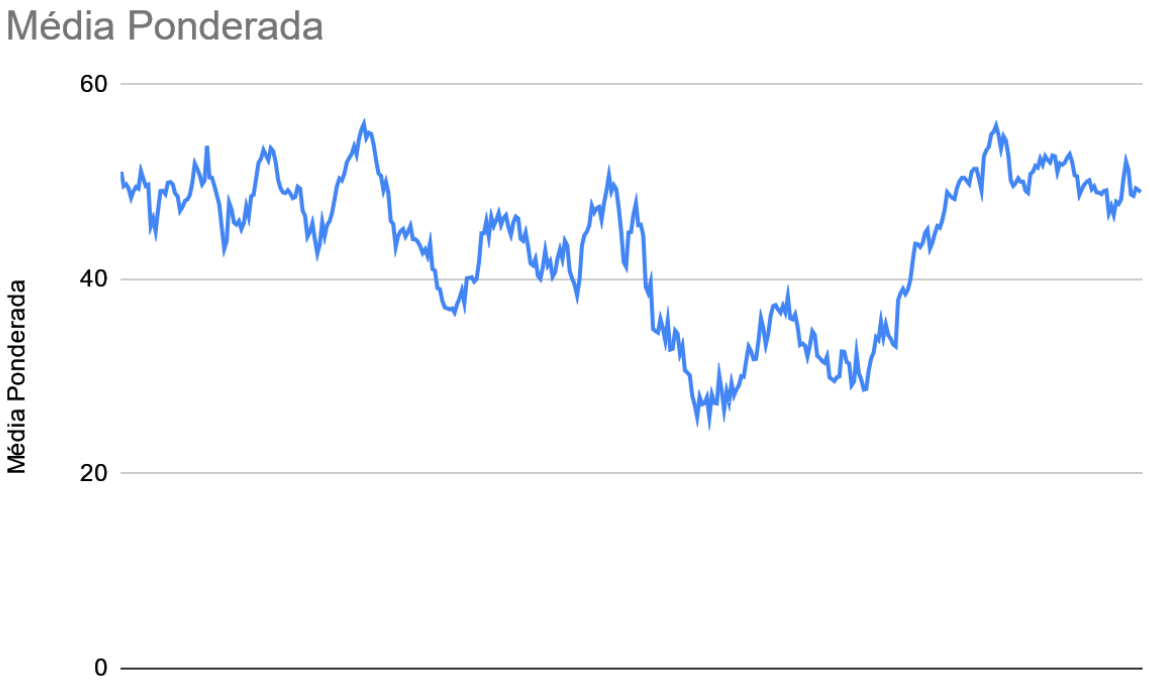
4. Plot do retorno vs. risco no período de testes



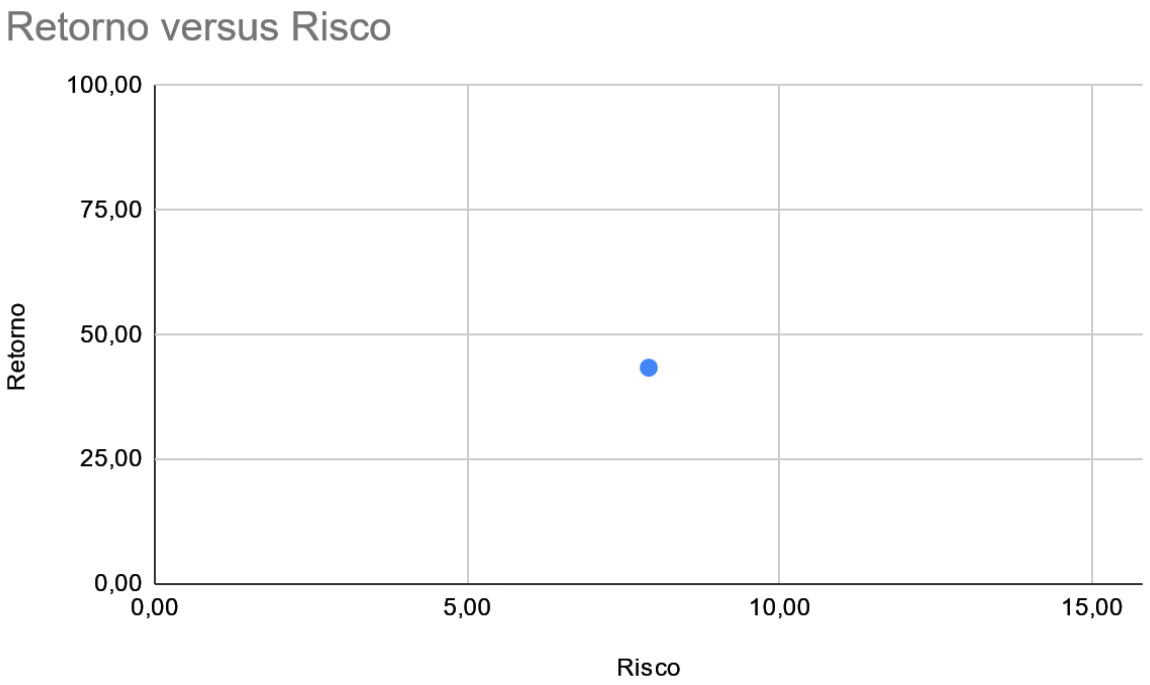
A Média apresentou um lucro de 51,22, o melhor resultado entre as outras técnicas, sua máxima de quase 80, com um risco de 11,11.

4.2.2 Média Ponderada

1. Retorno: 43,39
2. Risco: 7,90
3. Plot do retorno acumulado no período de testes



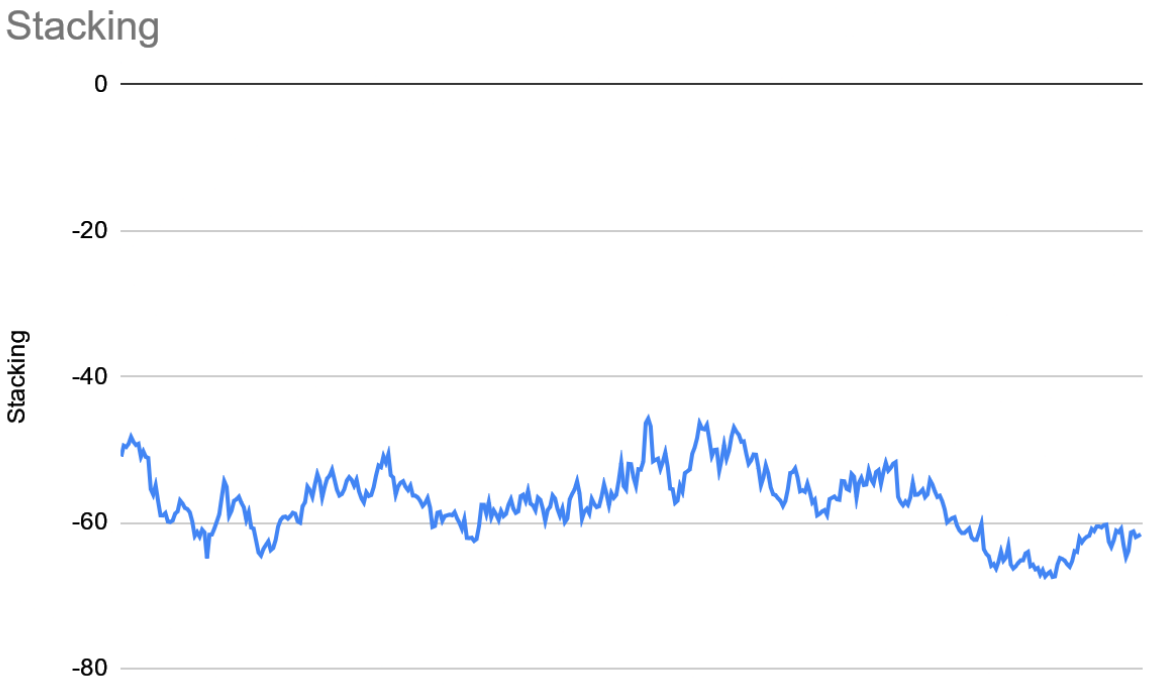
4. Plot do retorno vs. risco no período de testes



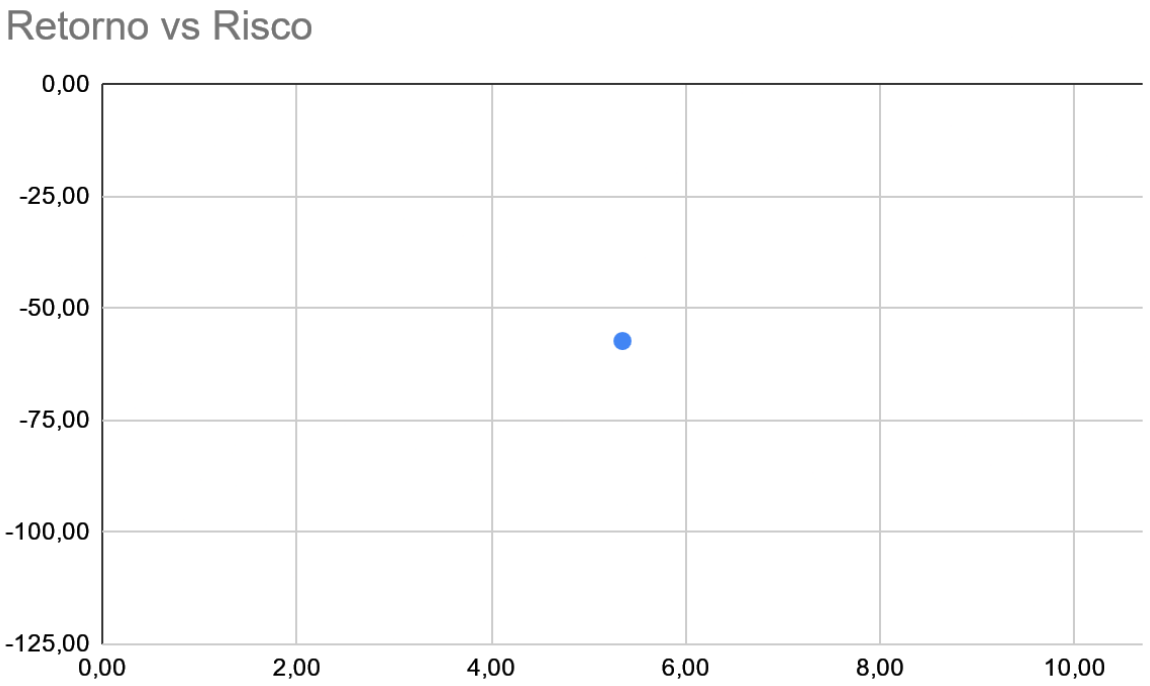
A Média Ponderada apresentou um lucro de 43,39, sua máxima foi mais de quase 60, com um risco de 7,90.

4.2.2 Stacking

1. Retorno: -57,24
2. Risco: 5,35
3. Plot do retorno acumulado no período de testes



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



O Stacking apresentou um péssimo resultado com um prejuízo de -57,24, o maior de todo trabalho, sua máxima não chegou a nem ser positiva, com o maior risco de 5,35.