MO432- Atividade 3

RA234837 - Bruno César de Oliveira Souza

RA065068 - Walker Humberto Batista

Objetivo

A atividade consiste em fazer previsões em series temporais usando as técnicas de aprendizado de máquina vistas. A série temporal utilizada contém o preço semanal do ouro em reais, de 18/06/2000 até 13/06/2021. As 100 das entradas mais recentes serão utilizadas como conjunto de treinamento.

O desafio é dividido em duas frentes de trabalho – regressão e classificação.

Para a regressão deve-se buscar minimizar o *RMSE* sobre o conjunto de treinamento. Já a classificação a maior acuracidade.

Neste trabalho seguimos buscamos explorar a comparação das diversas técnicas aprendidas durante a disciplinas e compará-las com algumas implementações de *deep-learning*.

Regressão

No que tange a parte de regressão implementamos as seguintes técnicas:

Machine Learning:

1. Ridge
2. Lasso
3. Máquina de suporte (SVR)
4. K Vizinhos
5. Árvores de decisões
6. Random Forest
7. Gradient Boosting Machine (GBM)
8. Voting Ensemble

Deep learning:

9. Redes Neurais Recorrentes
10. Redes Long short-term memory (LSTM)
11. Redes Residual LSTM
12. Redes Gated Recurrent Units

Parte – Machine Learning "Clássica"

Variáveis sintéticas

Para as técnicas de 1 a 8 criamos as seguintes variáveis sintéticas:

- **Derivação dos preços:** transformamos os preços absolutos em percentuais de oscilação, dados que existe uma influência exponencial nos dados que pode ser explicado pelo fator inflacionário do preço. Tornando a oscilação do dia seguinte o alvo de nossa previsão.
- **Janela deslizante:** utilizamos a função shift do Pandas para atrasar as oscilações em uma determinada quantidade de semanas. Neste trabalho exploramos esta janela de atraso no intervalo 1 a 52. Pela amostra de dados ser realizada semanalmente, supomos que um ano de atraso fosse um limite razoável para influência no futuro dos preços e, portanto, 52 como limite superior.
- **Número da semana:** para cada data foi associado o seu respectivo número da semana do ano e realizado um *one-hot-encode* para auxiliar em reconhecimento de possíveis padrões sazonais anuais.
- **Número do mês:** para cada data foi associado o seu respectivo número do mês e realizado um *one-hot-encode* para auxiliar em reconhecimento de possíveis padrões sazonais mensais.
- **Ano:** para cada data foi associado o seu respectivo ano e realizado um *one-hot-encode* para auxiliar em reconhecimento de possíveis tendências anuais.

Tratamento dos dados

Depois de criado as variáveis de suporte para o modelo, centralizamos e normalizamos os dados via *StandardScaler* do *SKLearn*.

Escolha de atributos

Até este ponto estávamos com um conjunto de 139 atributos potencialmente explicativo para nossos regressores. Utilizamos a regressão por mínimos quadrados parciais (PLS) do SKLearn para realizarmos a redução de dimensão do conjunto de nossos atributos. Como resultado tivemos que o melhor tamanho conjunto de componentes foi 1. Analisando com cuidado o resultado da PLS notamos que a melhor previsão utiliza como base o valor de oscilação do último dia.

Otimização de hiperparâmetros

Para cada técnica utilizada realizamos uma *grid search* com as seguintes parametrizações para cada método:

```
'Ridge':{
  'alpha': np.power(10, np.arange(-3,4, dtype=float))
},
'Lasso':{
  'alpha': np.power(10, np.arange(-3,4, dtype=float))
},
'SVR':{
  'kernel': ['linear', 'rbf'],
  'epsilon': [0.001, 0.01, 0.1],
  'C': np.power(2, np.arange(-5,15, dtype=float)),
  'gamma': np.power(2, np.arange(-9,3, dtype=float)),
  'max_iter': [1000]
},
'KNeighborsRegressor':{
  'n_neighbors': np.arange(1,10)
},
'DecisionTreeRegressor':{
  'ccp_alpha': np.arange(0, 0.044, 0.004, dtype=float)
},
'RandomForestRegressor':{
  'n_estimators': [10,100,1000]
},
'GradientBoostingRegressor':{
  'learning_rate': [0.01,0.3],
  'max_depth': [2,3]
}
```

Ensemble

Em experimento para melhorar a robustez da previsão testamos algumas técnicas de *ensemble* implementadas no pacote SKLearn: Random Forest, Gradient Boosting Machine e Voting com a combinação dos métodos de 1 a 7.

Parte – Deep Learning

Utilizamos uma estratégia de separar o projeto em modelos de machine learning tradicional e deep learning. A primeira parte necessitava de um prévio tratamento dos dados (pre-processing) antes de alimentar os modelos de previsão de regressão. Com o intuito de uma abordagem comparativa aplicamos modelos de deep learning que aplica um aprendizado de representatividade dos dados sem um pré processamento de modo elaborado.

Foi utilizado o framework TensorFlow programado em Python.

Variáveis sintéticas

Para alimentar o modelo, criamos uma classe denominada WindowGenerator(). Essa classe tinha a função de gerar, de modo flexível, uma variedade de **janelas deslizante de dados consecutivos**.

```python
class WindowGenerator():
  def __init__(self, input_width, label_width, shift,
               train_df=train_df, val_df=val_df, test_df=test_df,
               label_columns=None):
    # Store the raw data.
    self.train_df = train_df
    self.val_df = val_df
    self.test_df = test_df

    # Work out the label column indices.
    self.label_columns = label_columns
    if label_columns is not None:
      self.label_columns_indices = {name: i for i, name in
                                    enumerate(label_columns)}
    self.column_indices = {name: i for i, name in
                           enumerate(train_df.columns)}

    # Work out the window parameters.
    self.input_width = input_width
    self.label_width = label_width
    self.shift = shift

    self.total_window_size = input_width + shift

    self.input_slice = slice(0, input_width)
    self.input_indices = np.arange(self.total_window_size)[self.input_slice]

    self.label_start = self.total_window_size - self.label_width
    self.labels_slice = slice(self.label_start, None)
    self.label_indices = np.arange(self.total_window_size)[self.labels_slice]
```

Figure 1 - Representação parcial da classe WindowGenerator

A Figura abaixo ilustra um exemplo de instância da classe em que o tamanho da janela é 6 e a previsão é relacionado ao dado posterior da janela.

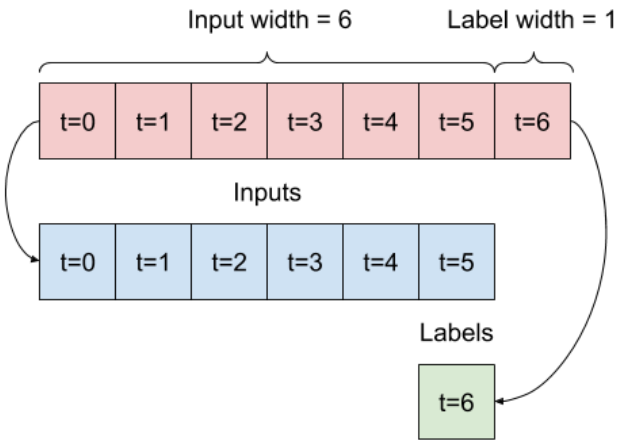


Figure 2 - Ilustração dos dados de entrada e label do modelo de deep learning

Importante ressaltar que o modelo recebe o “inputs” e tenta prever o “label”, sem nenhum pré-processamento da variável.

Tratamento dos dados

Realizamos a normalização de todos os dados entre 0 e 1 via *MinMaxScaler* do *SKLearn*.

Resultados

Técnica Regressão	RMSE	Hiperparâmetros
Ridge	45,4534	{'alpha': 0.001}
Lasso	45,5374	{'alpha': 100.0}
SVR	45,1676	{'C': 0.125, 'epsilon': 0.01, 'gamma': 0.001953125, 'kernel': 'rbf', 'max_iter': 1000}
K-Neighbors	45,8837	{'n_neighbors': 9}
Decision Tree	43,7622	{'ccp_alpha': 0.004}
Random Forest	51,9839	{'n_estimators': 1000}
Gradient Boosting Machine	44,5165	{'learning_rate': 0.01, 'max_depth': 3}
Voting Ensemble	44,8282	-
LSTM	125,7032	32 camadas LSTM sequenciais
Residual LSTM	29,5401	32 camadas LSTM residuais
RNN	104,0853	32 camadas RNN sequenciais
GRU	61,4537	32 camadas GRU sequenciais

Em nosso experimento notamos que na média as técnicas de *machine learning* clássicas apresentaram boa performance na média em comparação as de *deep learning*, mas exigiram maior esforço no tratamento dos dados. Apesar do melhor resultado ter sido atingido pela técnica de *Residual LSTM* com o RMSE de 29,5401 os resultados foram muito semelhantes as técnicas clássicas, como podemos observar nas figuras abaixo.

Uma importante ressalve deve ser feito em relação ao tamanho do dados de treinamento. Modelos de *deep learning* necessitam de um grande conjunto de dados de treinamento para apresentar boa performance, portanto, essa questão pode explicar parcialmente o fato desses modelos apresentarem resultados modestos quando comparados aos modelos de *machine learning* clássicos.

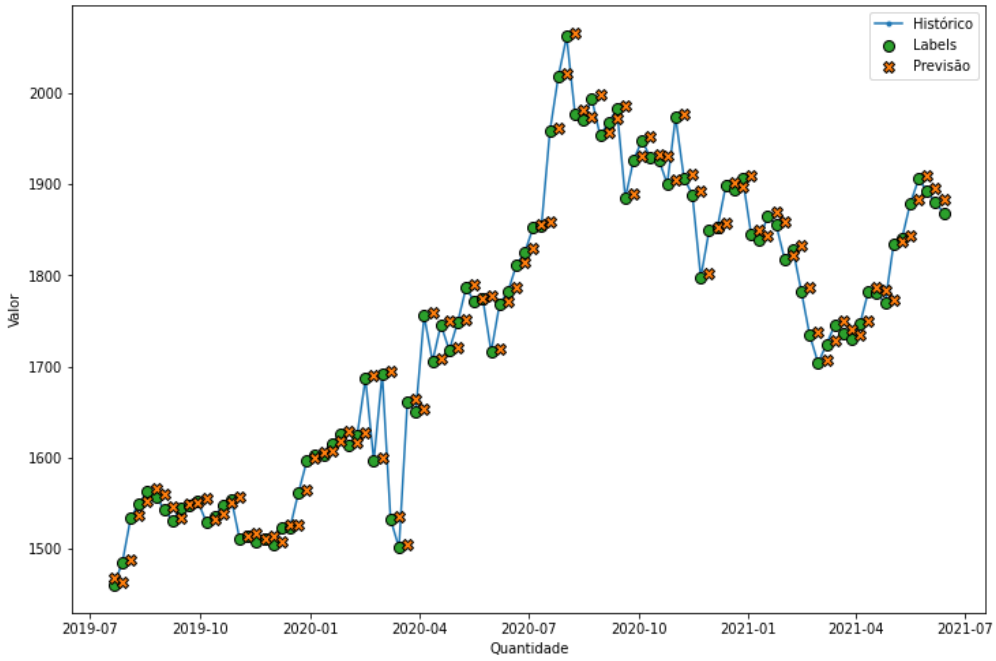


Figure 3 - Regressão Decision Tree

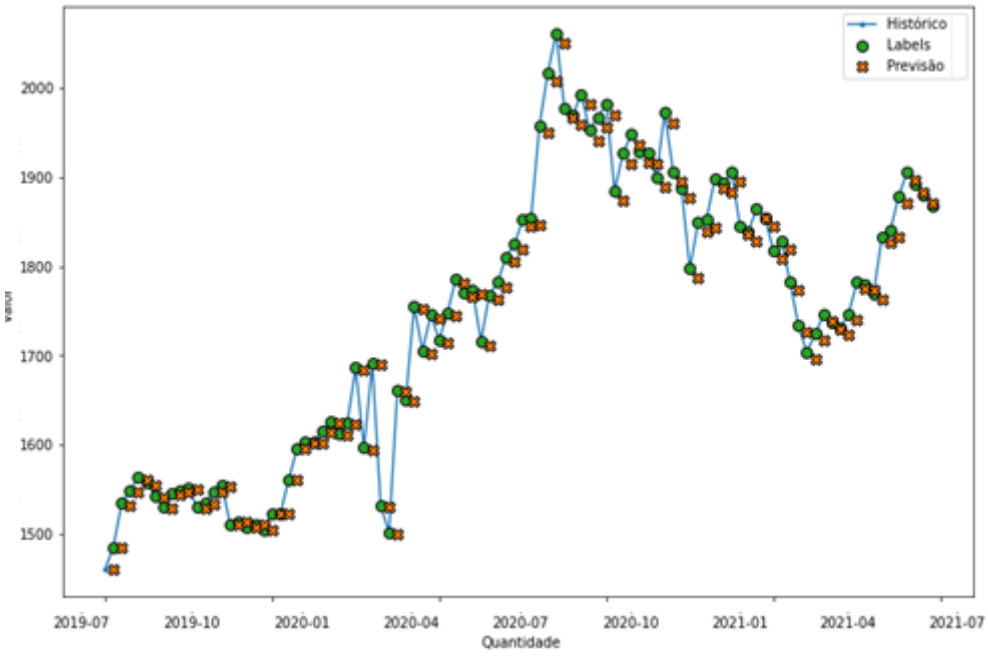


Figure 4 - Regressão Residual LSTM

Classificação

Na parte de classificação implementamos as seguintes técnicas:

Machine learning:

1. Elastic Net
2. Máquina de suporte (SVC)
3. K Vizinhos
4. Árvores de decisões
5. Random Forest
6. Gradient Boosting Machine (GBM)

Deep learning:

7. Redes Neurais Recorrentes
8. Redes Long short-term memory (LSTM)
9. Redes Residual LSTM
10. Redes Gated Recurrent Units

Parte – Machine Learning "Clássica"

Variáveis sintéticas

Para as técnicas de 1 a 6 criamos as seguintes variáveis sintéticas:

- **Derivação dos preços:** transformamos os preços absolutos em percentuais de oscilação, dados que existe uma influência exponencial nos dados que pode ser explicado pelo fator inflacionário do preço. Depois disso, alteramos a oscilação em duas classes, ALTA (1) e BAIXA (0), e este é o alvo de nossa classificação.
- **Janela deslizante:** utilizamos a função shift do Pandas para atrasar as oscilações em uma determinada quantidade de semanas. Neste trabalho exploramos esta janela de atraso no intervalo 1 a 52. Pela amostra de dados ser realizada semanalmente, supomos que um ano de atraso fosse um limite razoável para influência no futuro dos preços e, portanto, 52 como limite superior.
- **Número da semana:** para cada data foi associado o seu respectivo número da semana do ano e realizado um *one-hot-encode* para auxiliar em reconhecimento de possíveis padrões sazonais anuais.
- **Número do mês:** para cada data foi associado o seu respectivo número do mês e realizado um *one-hot-encode* para auxiliar em reconhecimento de possíveis padrões sazonais mensais.
- **Ano:** para cada data foi associado o seu respectivo ano e realizado um *one-hot-encode* para auxiliar em reconhecimento de possíveis tendências anuais.

Tratamento dos dados

Depois de criado as variáveis de suporte para o modelo, para facilitar o algoritmo de treinamento, deixamos todos os dados entre 0 e 1 via *MinMaxScaler* do *SKLearn*.

Escolha de atributos

Até este ponto estávamos com um conjunto de 139 atributos potencialmente explicativo para nossos regressores. Utilizamos seleção por percentil, utilizando a métrica F-value da tabela ANOVA com o percentil igual a 10. Como resultado tivemos que o melhor tamanho conjunto de componentes foi 14.

Otimização de hiperparâmetros

Para cada técnica utilizada realizamos uma *grid search* com as seguintes parametrizações para cada método:

```
params ={
  'ElasticNet':{
    'alpha': np.power(10, np.arange(-3,4, dtype=float)),
    'l1_ratio': np.arange(0.001, 0.8, 0.001, dtype=float)
  },
  'SVC':{
    'kernel': ['rbf'],
    'C': np.power(2, np.arange(-5,15, dtype=float)),
    'gamma': np.power(2, np.arange(-9,3, dtype=float)),
    'max_iter': [10000]
  },
  'KNeighborsClassifier':{
    'n_neighbors': np.arange(1,10)
  },
  'DecisionTreeClassifier':{
    'ccp_alpha': np.arange(0, 0.044, 0.004, dtype=float),
    'class_weight': [None, 'balanced']
  },
  'RandomForestClassifier':{
    'n_estimators': [10,100,1000],
    'class_weight': [None, 'balanced']
  },
  'GradientBoostingClassifier':{
    'learning_rate': [0.01,0.3],
    'max_depth': [2,3]
  }
}
```

Ensemble

Em experimento para melhorar a robustez da previsão testamos algumas técnicas de *ensemble* implementadas no pacote SKLearn: Random Forest e Gradient Boosting Machine.

Parte – Deep Learning

Variáveis sintéticas

Da mesma forma que a etapa de regressão foi utilizada a classe WindowsGenerator(). Entretanto para esse etapa criamos a seguinte variável sintética:

- **Variação dos preços:** Criamos uma coluna nos dados que estipula a oscilação do ouro semanalmente. Alteramos a oscilação em duas classes, ALTA (1) e BAIXA (0), e este é o alvo de nossa classificação.

Dessa forma, o modelo recebe como entrada a janela deslizante e realiza a previsão sobre a coluna de variação do preço.

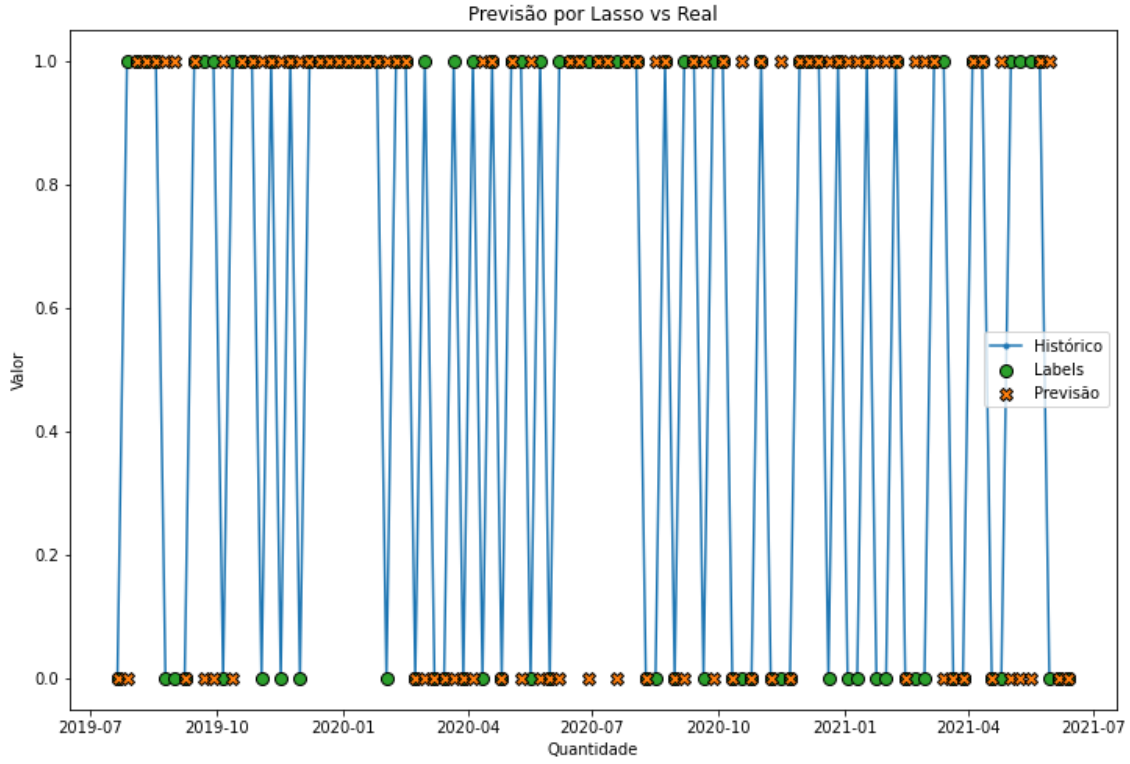
Tratamento dos dados

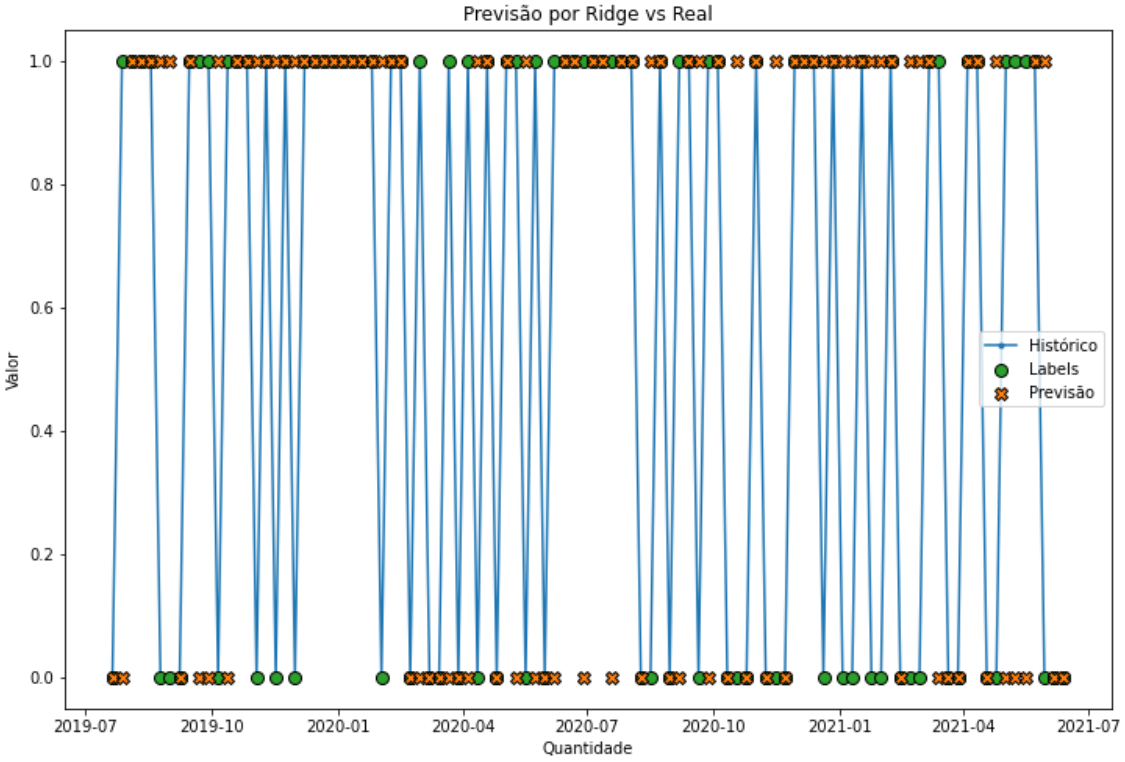
Realizamos a normalização de todos os dados entre 0 e 1 via *MinMaxScaler* do *SKLearn*.

Resultados

Técnica Classificação	Acuraciadade	Hiperparâmetros
Elastic Net	58%	{'alpha': 0.001, 'l1_ratio': 0.001}
SVC	54%	{'C': 0.5, 'gamma': 2.0, 'kernel': 'rbf', 'max_iter': 10000}
K-Neighbors	53%	{'n_neighbors': 7}
Decision Tree	58%	{'ccp_alpha': 0.008 , 'class_weight': None}
Random Forest	51%	{'n_estimators': 1000 , 'class_weight': None}
Gradient Boosting Machine	58%	{'learning_rate': 0.01, 'max_depth': 2}
LSTM	49%	32 camadas LSTM sequenciais
Residual LSTM	52%	32 camadas LSTM residuais
RNN	52%	32 camadas RNN sequenciais
GRU	49%	32 camadas GRU sequenciais
Regressão para Classificação	60%	Lasso e Ridge

Em nosso experimento o melhor classificador para o caso da série de preços do ouro foi a transformação do resultado da regressão Lasso ou Ridge em classes de alta e baixa, com acuracidade de 60%. Seguem os gráficos dos dois melhores classificadores.




Previsão por Ridge vs Real
Valor
Quantidade
Histórico
Labels
Previsão
1.0
0.8
0.6
0.4
0.2
0.0
2019-07
2019-10
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04
2021-07