Documentação Técnica: Pipeline de COFDI-ERP

Análise de Investimentos Chineses

1 Introdução Técnica

Este documento detalha a implementação técnica completa da pipeline de análise dos Investimentos Chineses no Exterior (COFDI) e seu relacionamento com a política ERP. O código emprega uma abordagem sistemática dividida em 11 etapas, utilizando bibliotecas especializadas para cada finalidade.

2 Arquitetura de Bibliotecas e Justificativas Técnicas

2.1 Configuração e Utilitários Básicos

- **matplotlib.use('Agg')**: Configuração para geração de gráficos em ambiente servidor
- **warnings.filterwarnings('ignore')**: Remove alertas para outputs mais limpos
- **pathlib.Path**: Manipulação de caminhos de arquivo multiplataforma
- **pd.set_option()**: Configuração do pandas para melhor visualização de dados

2.2 Processamento de Dados

- **pandas**: Manipulação eficiente de DataFrames com dados tabulares
- **numpy**: Operações numéricas vetorizadas para cálculos rápidos
- **re**: Expressões regulares para limpeza de strings numéricas complexas
- **csv**: Parser alternativo para dados com formatação irregular

2.3 Visualização e Gráficos

- **matplotlib.pyplot**: Criação de gráficos estáticos personalizáveis
- **seaborn**: Visualizações estatísticas com temas pré-definidos
- **plotly.graph_objects**: Gráficos interativos e diagramas sunburst
- **geopandas**: Criação de mapas coropléticos e análise geoespacial
- **matplotlib.ticker**: Formatação personalizada de eixos numéricos

2.4 Análise Estatística e Econométrica

- **statsmodels.api**: Modelos estatísticos tradicionais (OLS)
- **statsmodels.tsa**: Modelagem de séries temporais (ARIMA, Markov Switching)
- **pingouin**: Implementação de testes estatísticos com estilo R
- **ruptures**: Detecção de pontos de mudança estrutural em séries

2.5 Machine Learning e Pré-processamento

- **sklearn.pipeline**: Encadeamento de transformações e modelos
- **sklearn.preprocessing**: Normalização, codificação e imputação de dados
- **sklearn.model_selection**: Validação cruzada temporal e estratificada
- **sklearn.ensemble**: Random Forest para modelos baseline
- **sklearn.metrics**: Métricas abrangentes de avaliação de modelos

2.6 Algoritmos Avançados de ML

- **xgboost**: Gradient Boosting com otimizações de performance
- **lightgbm**: Gradient Boosting com uso eficiente de memória
- **catboost**: Processamento nativo de variáveis categóricas
- **prophet**: Modelagem de séries temporais com decomposição sazonal

2.7 Econometria de Machine Learning

- **econml.dml**: Double Machine Learning para inferência causal
- **econml.metalearners**: Meta-learners para efeitos de tratamento heterogêneos
- **shap**: Interpretabilidade de modelos através de valores Shapley

3 Implementação das 11 Etapas

3.1 ETAPA 1: Configuração e Estrutura de Diretórios

Bibliotecas principais: os, pathlib

```
DIRS = {
    "BASE": RESULTS_DIR, "R_STYLE": R_STYLE_DIR,
    "EDA": RESULTS_DIR / 'eda', "PHASE_ANALYSIS": RESULTS_DIR / 'phase_analysis',
    "REGRESSION": RESULTS_DIR / 'models' / 'regression',
    "CLASSIFICATION": RESULTS_DIR / 'models' / 'classification',
    "TIMESERIES": RESULTS_DIR / 'models' / 'timeseries',
    "CAUSAL": RESULTS_DIR / 'models' / 'causal', "SHAP": RESULTS_DIR / 'models' / 'shap'
}
```

Objetivo: Organização sistemática de todos os resultados para fácil localização e reprodução.

3.2 ETAPA 2: Carregamento e Limpeza de Dados

Bibliotecas: pandas, re, csv, numpy

```
def clean_numeric_column(series: pd.Series) -> pd.Series:
    cleaned = series.astype(str).str.replace(r'[\[\]$,%\s]', '', regex=True)
    numeric_direct = pd.to_numeric(cleaned, errors='coerce')
    # Método alternativo com regex para casos complexos
    NUMBER_REGEX = r'([-+]?[0-9]*\.?[0-9]+(?:[eE][-+]?[0-9]+)?)'
```

Característica: Sistema de limpeza em duas etapas que combina conversão direta com expressões regulares para lidar com diferentes formatos numéricos.

3.3 ETAPA 3: Engenharia de Features

Bibliotecas: pandas, numpy

```
# Criação de features temporais avançadas
for window in [2, 3]:
    df[f'valor_roll_mean_{window}'] = df.groupby('Sector')['Valor_USD']\
        .transform(lambda x: x.rolling(window, min_periods=1).mean())
```

Features desenvolvidas:

- **Médias móveis setoriais**: Capturam tendências e momentum por setor
- **Variáveis de política**: `post_ERP`, `policy_cumulative` para análise temporal
- **Features hierárquicas**: Combinação ponderada de médias setoriais e globais
- **Targets adaptativos**: Classificação baseada em percentis anuais

3.4 ETAPA 1.5: Visualizações Estilo R

Bibliotecas: matplotlib, seaborn, geopandas, plotly, joypy

```
def plot_ggstats_style(df_plot, x_var, y_var, output_dir, filename, title_prefix):
    # Combinação de violin plots, box plots e stripplots
    # Inclusão de anotações estatísticas automáticas
    stats = pg.ttest(df_plot[y_var], df_plot[x_var], correction='auto')
```

Característica: Recriação do estilo visual do R com:

- Testes estatísticos incorporados diretamente nos gráficos
- Anotações automáticas de médias e valores extremos
- Formatação matemática com LaTeX nas legendas

3.5 ETAPA 4: Análise Exploratória (EDA)

Bibliotecas: pandas, matplotlib, seaborn

```
# Análise de mudanças setoriais antes e depois da ERP
sector_share = sector_summary.apply(lambda x: 100 * x / x.sum(), axis=0)
sector_share['Change (p.p.)'] = sector_share['Post-ERP Share (%)'] - sector_share['Pre-ERP Share (%)
```

3.6 ETAPA 4.5: Análise por Fase GG

Bibliotecas: statsmodels, pandas

```
def run_ols_interaction_phase(df_phase, phase_name, y_var, x_var, z_var, results_dir):
    # Modelos de interação: Y ~ X * C(Z)
    formula = f"Q('{y_var}') ~ Q('{x_var}') * ({' + '.join(dummy_cols)})"
    model = ols(formula, data=df_temp).fit()
```

Abordagem: Modelos separados para cada fase "Going Global"para capturar variações temporais.

3.7 ETAPA 5: Pipelines de ML

Bibliotecas: sklearn.pipeline, sklearn.compose, sklearn.preprocessing

```
preprocessor_linear = ColumnTransformer([
    ('num', numeric_transformer_linear, numeric_features),
    ('cat', categorical_transformer_ohe, categorical_features)
])
```

Arquitetura: Dois pipelines especializados para diferentes tipos de modelo:

- **Linear**: OneHotEncoder para modelos lineares
- **Tree**: OrdinalEncoder para modelos baseados em árvores

3.8 ETAPA 6: Modelos de Regressão Otimizados

Bibliotecas: xgboost, lightgbm, sklearn.model_selection

```
def mape_scorer(y_log, y_pred_log):
    y_orig = np.exp(y_log)
    y_pred_orig = np.exp(y_pred_log)
    mape = np.mean(np.abs((y_orig - y_pred_orig) / np.maximum(y_orig, 1))) * 100
    return -mape
```

Característica: Métrica MAPE adaptada que:

- Opera em escala logarítmica durante o treinamento
- Converte para escala original para avaliação final
- Previne divisão por zero através de `np.maximum(y_orig, 1)`

3.9 ETAPA 7: Modelos de Classificação

Bibliotecas: xgboost, lightgbm, catboost, sklearn.metrics

```
# Análise de calibração e curvas Precision-Recall
fraction_pos, mean_pred = calibration_curve(y_test, y_proba, n_bins=10)
precision, recall, _ = precision_recall_curve(y_test, y_proba)
```

3.10 ETAPA 8: Séries Temporais

Bibliotecas: statsmodels.tsa, prophet

```
# ARIMA para modelagem tradicional
model_arima = ARIMA(ts_log, order=(1, 1, 1))
# Prophet para modelagem com componentes
m = Prophet()
m.fit(prophet_df)
```

Abordagem: Combinação de métodos tradicionais (ARIMA) e modernos (Prophet).

3.11 ETAPA 9: Modelos Causais

Bibliotecas: econml, statsmodels.tsa

```
# Double Machine Learning para estimativa causal
dml_est = LinearDML(model_y=model_y, model_t=model_t, discrete_treatment=True)
dml_est.fit(Y_causal, T_causal, X=X_processed, W=None)

# Markov Switching para identificação de regimes
mod_ms = MarkovRegression(endog=ts_data['Valor_USD'], k_regimes=2, switching_variance=True)
```

Metodologia: DML fornece estimativas de efeito causal controlando por variáveis de confusão através de machine learning.

3.12 ETAPA 10: Interpretabilidade SHAP

Bibliotecas: shap, sklearn

```
explainer = shap.TreeExplainer(fitted_model)
shap_values = explainer(shap_input_data)
shap.summary_plot(shap_values, shap_input_data, plot_type="dot")
```

Utilidade: Identifica e visualiza o impacto de cada variável nas previsões do modelo.

4 Características Técnicas Implementadas

4.1 Pré-processamento de Dados

- **Sistema de parsing duplo**: CSV nativo com fallback manual para dados problemáticos
- **Limpeza numérica abrangente**: Combinação de métodos diretos e expressões regulares
- **Tratamento de alta cardinalidade**: Estratégias diferenciadas para variáveis categóricas

4.2 Validação de Modelos

- **TimeSeriesSplit**: Preserva a ordem temporal durante a validação
- **StratifiedKFold**: Mantém a distribuição das classes em problemas de classificação
- **Métricas específicas**: MAPE adaptado para dados econômicos

4.3 Arquitetura do Sistema

- **Pipelines reproduzíveis**: Todo o pré-processamento encapsulado e versionado
- **Tratamento de exceções**: Continuação da execução mesmo com falhas em componentes individuais
- **Logging detalhado**: Monitoramento do status de cada etapa do processo

4.4 Visualização

- **Estilo acadêmico**: Gráficos com qualidade para publicação científica
- **Anotações estatísticas**: Testes de hipótese integrados nas visualizações
- **Análise geoespacial**: Mapas interativos para distribuição geográfica

5 Conclusão

Esta implementação constitui uma pipeline completa para análise de dados econômicos, integrando métodos econométricos tradicionais com técnicas sofisticadas de machine learning. A arquitetura modular permite reprodução precisa dos resultados e adaptabilidade para análises futuras.