# Sistema de Manutenção Preditiva - Bootcamp CDIA

Este projeto implementa um sistema inteligente de manutenção preditiva para identificar falhas em máquinas industriais usando Machine Learning.

## Executando com Docker

### Pré-requisitos

- Docker instalado
- Docker Compose instalado

### Estrutura do Projeto

```
projeto/
├── Dockerfile
├── docker-compose.yml
├── requirements.txt
├── main.py
├── RandomForest.ipynb
├── data/
│   ├── bootcamp_train.csv
│   └── bootcamp_test.csv (opcional)
├── models/
├── outputs/
├── visualizations/
└── notebooks/
```

### Configuração Inicial

1. **Clone/baixe o projeto e organize os arquivos:**

```bash
mkdir bootcamp_ml_project
cd bootcamp_ml_project
```

2. **Coloque seus arquivos de dados na pasta `data/`:**

```bash
mkdir data
# Copie bootcamp_train.csv para data/
# Copie bootcamp_test.csv para data/ (se disponível)
```

3. **Crie as pastas necessárias:**

```bash
mkdir models outputs visualizations notebooks
```

## Execução com Docker

### Opção 1: Usando Docker Compose (Recomendado)

```bash
# Construir e iniciar o container
docker-compose up --build

# Ou para executar em background
docker-compose up -d --build
```

### Opção 2: Usando Docker diretamente

```bash
# Construir a imagem
docker build -t bootcamp-ml .

# Executar o container
docker run -p 8888:8888 -v $(pwd)/data:/app/data -v $(pwd)/outputs:/app/outputs bootcamp-ml
```

## Acessando o Jupyter Notebook

Após executar o container, acesse:

- **URL:** http://localhost:8888
- **Token:** Não é necessário (configurado para acesso direto)

## Executando o Script Python Principal

Para executar o pipeline completo via script Python:

```bash
# Entrar no container
docker exec -it bootcamp_ml_project bash

# Executar o script principal
python main.py
```

## Estrutura dos Dados

O sistema espera os seguintes arquivos na pasta `data/`:

- `bootcamp_train.csv` - Dados de treinamento
- `bootcamp_test.csv` - Dados de teste (opcional)

## Outputs Gerados

O sistema gera os seguintes arquivos:

- `models/modelo_otimizado.pkl` - Modelo treinado
- `outputs/submission.csv` - Arquivo de submissão final
- `visualizations/` - Gráficos e análises visuais

## Funcionalidades

1. **Diagnóstico completo dos dados**
2. **Limpeza e pré-processamento**
3. **Análise exploratória com visualizações**
4. **Treinamento com otimização de hiperparâmetros**
5. **Avaliação do modelo**
6. **Geração de predições**

## Comandos Úteis

```bash
# Parar os containers
docker-compose down

# Ver logs
docker-compose logs

# Limpar tudo
docker-compose down --volumes --rmi all

# Entrar no container em execução
docker exec -it bootcamp_ml_project bash
```

## Customização

Para modificar as configurações:

1. **Hiperparâmetros:** Edite `param_grid` em `main.py`

2. **Portas:** Modifique `docker-compose.yml`
3. **Dependências:** Atualize `requirements.txt`

## Troubleshooting

**Problema:** Porta 8888 já em uso

```bash
# Usar porta diferente
docker run -p 8889:8888 bootcamp-ml
```

**Problema:** Permissões de arquivo

```bash
# No Linux/Mac, ajustar permissões
sudo chown -R $USER:$USER ./data ./outputs
```

**Problema:** Falta de memória

```bash
# Limitar recursos do container
docker run -m 2g bootcamp-ml
```

## Desenvolvimento

Para desenvolver e modificar o código:

1. Monte o código como volume:

```bash
docker run -v $(pwd):/app -p 8888:8888 bootcamp-ml
```

2. Use Jupyter Lab (mais moderno):

```bash
# Modificar CMD no Dockerfile para:
CMD ["jupyter", "lab", "--ip=0.0.0.0", "--port=8888", "--no-browser", "--allow-root"]
```

## Próximos Passos

- [ ] Implementar API REST com FastAPI
- [ ] Criar dashboard com Streamlit

- [ ] Configurar CI/CD
- [ ] Deploy na nuvem
- [ ] Monitoramento MLOps