Detecção de fraude em transações de cartão de crédito

Lucas Oliveira

Ordem da apresentação

Seção 1:

Definição do Case

Seção 2:

Dados utilizados

Seção 3:

Análise Exploratória

Seção 4:

Pré-Processamento

Seção 5:

Escolha do classificador

Seção 6:

Otimização dos hiperparâmetros

Seção 7:

Avaliação dos resultados

Seção 8:

Conclusão

Definição do Case

- Detectar fraudes em transações de cartões de crédito.
- Construir um modelo de classificação.
- Transações de cartão de crédito realizadas durante 2 dias.
- Dados muito desbalanceados.

Base de dados

Informações

- 284.807 transações.
- 31 variáveis.

Variáveis

- **Time**: Segundos decorridos entre esta transação e a primeira transação no conjunto de dados.
- **V1 - V28**: Variáveis afetadas pela aplicação do método de redução de dimensionalidade PCA.
- **Amount**: Valor da transação.
- **Class**: Classificação da transação, sendo **1** para fraudulenta e **0** para não fraudulenta.

Análise exploratória

Transações

- **Nulos:** Não existem valores nulos.
- **PCA:** Variáveis V1-V28 estão dentro de uma escala.
- **Outliers:** Existem valores que podem ser considerados outliers.
- **Média:** US$ 88.34, mas esse valor é afetado pelos outliers.
- **99%:** O maior valor para 99% das transações é de US$ 1017.97, enquanto para 100% é de US$ 25691.16.
- **Fraudes:** 99.83% das transações são legítimas e somente 0.17% são fraudes.
- **Maiores valores:** As transações de maior valor estão classificadas como não fraudulentas.

Pré-processamento

Escalonamento de variáveis

- Time e Amount
- RobustScaler

Balanceamento das classes

- Undersampling (resample) e Oversampling (SMOTE)
- Divisão dos dados feita antes do balanceamento

Datasets balanceados

- Original: 284.807 transações.
- Undersampling: 766 transações
- Oversampling: 454.924 transações

Escolha do classificador

Validação Cruzada Estratificada

- StratifiedKFold
- Aplicado nos dados de Undersampling

Algoritmos

- Regressão Logística
- Support Vector Machine
- Gaussian Naive Bayes
- Random Forest
- Gradient Boosting
- Cat Boost
- XGBoost
- **LightGBM - Escolhido!**

ALGORITMO	MÉDIA_ACC	STD_ACC
CatBoost	0.942	0.022
LightGBM	0.942	0.018
Regressão Logística	0.941	0.024
XGBoost	0.937	0.020
Random Forest	0.932	0.021
Support Vector Machine	0.932	0.015

Tabela com os resultados da Validação Cruzada Estratificada

Otimização dos hiperparâmetros

Etapas de treinamento

- Undersampling
- Oversampling
- Validação final com dados originais

Divisão entre treino e teste

- Optuna para otimização
- Recall para avaliação

```python
# Definindo a função que o Optuna irá utilizar para testar os hiperparâmetros
def objective(trial):
    # Definindo os hiperparâmetros da LightGBM
    parametros = {
        'n_estimators': trial.suggest_int('n_estimators', 20, 200), # Quantidade máxima de nós
        'num_leaves': trial.suggest_int('num_leaves', 2, 50), # Quantidade máxima de folhas
        'max_depth': trial.suggest_int('max_depth', 3, 50), # Profundidade máxima das árvores
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.001, 0.2), # Taxa de aprendizado
        'subsample': trial.suggest_uniform('subsample', 0.6, 1.0), # Subsample usado para impedir overfitting
        'min_child_weight': trial.suggest_int('min_child_weight', 1, 20), # Quantidade mínima para criar um novo nó
        'lambda_l1': trial.suggest_loguniform('lambda_l1', 1e-6, 1.0), # Penalização Lasso nas árvores
        'lambda_l2': trial.suggest_loguniform('lambda_l2', 1e-6, 1.0), # Penalização Ridge nas árvores
    }

    # Criando o modelo LightGBM com os hiperparâmetros otimizados
    modelo = LGBMClassifier(**parametros, random_state=21)

    # Treinando o modelo nos dados já balanceados pelo Undersampling
    modelo.fit(X_train_undersampled, y_train_undersampled)

    # Previsões para a base de testes
    y_pred_undersampled = modelo.predict(X_test_undersampled)

    # Realiza o cálculo da métrica de Recall, que seria a métrica mais segura dadas as condições originais dos dados
    recall = recall_score(y_test_undersampled, y_pred_undersampled)

    return recall
```

Função utilizada pelo Optuna para encontrar os melhores parâmetros do modelo.

Modelos

Undersampling

```
LGBMClassifier
LGBMClassifier(boosting_type=LGBMClassifier(lambda_l1=2.013838835210629e-06,
                                            lambda_l2=0.0007771472931457983,
                                            learning_rate=0.007551211447829674,
                                            max_depth=41, min_child_weight=5,
                                            n_estimators=140, num_leaves=29,
                                            subsample=0.8750804543836551))
```

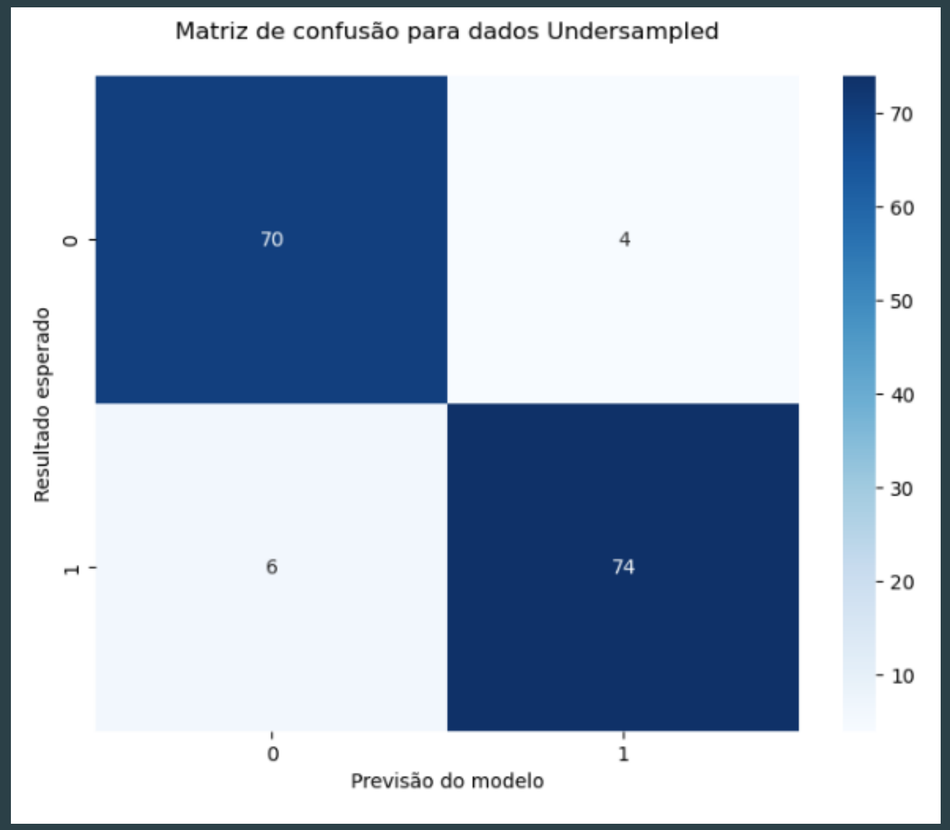
Oversampling

```
LGBMClassifier
LGBMClassifier(boosting_type=LGBMClassifier(lambda_l1=0.3828821458483766,
                                            lambda_l2=0.00045100317062221715,
                                            learning_rate=0.04926240577582323,
                                            max_depth=35, min_child_weight=8,
                                            n_estimators=181, num_leaves=37,
                                            subsample=0.7316863020085818))
```

Avaliação dos resultados

Undersampling

Matrix de confusão



Métricas de classificação

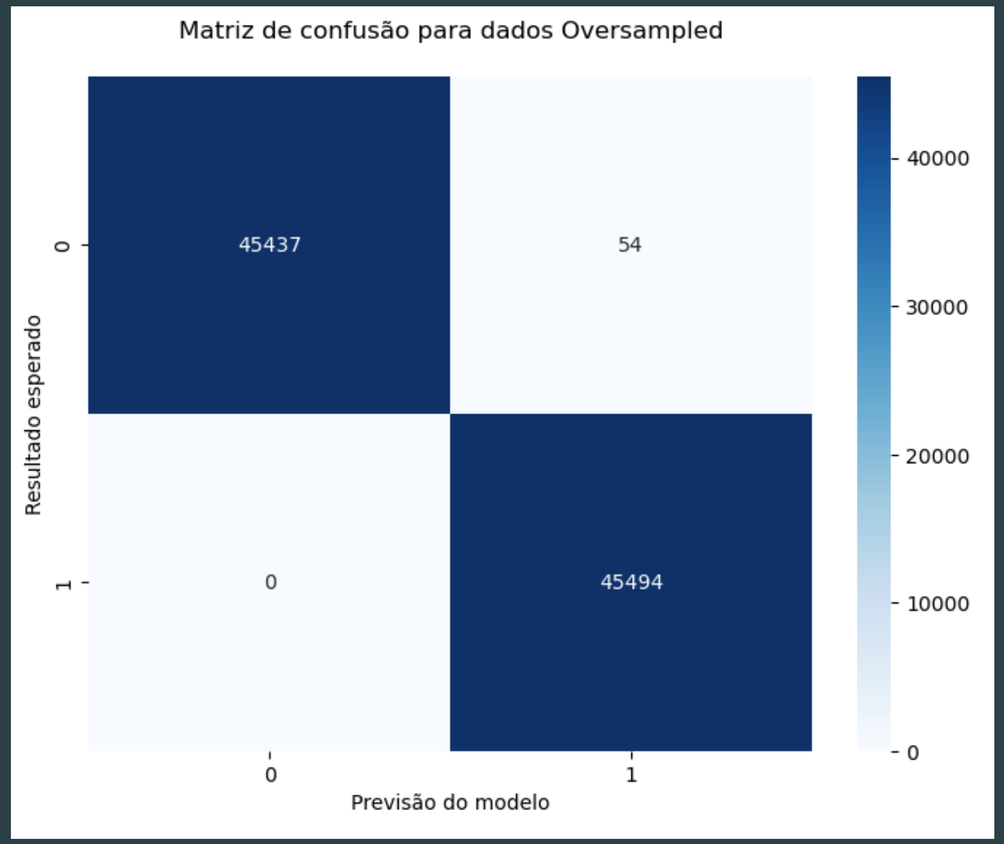
FRAUDE	PRECISION	RECALL	F1-SCORE
0	0.92	0.95	0.93
1	0.95	0.93	0.94

Acurácia: 0.94

Avaliação dos resultados

Oversampling

Matrix de confusão



Métricas de classificação

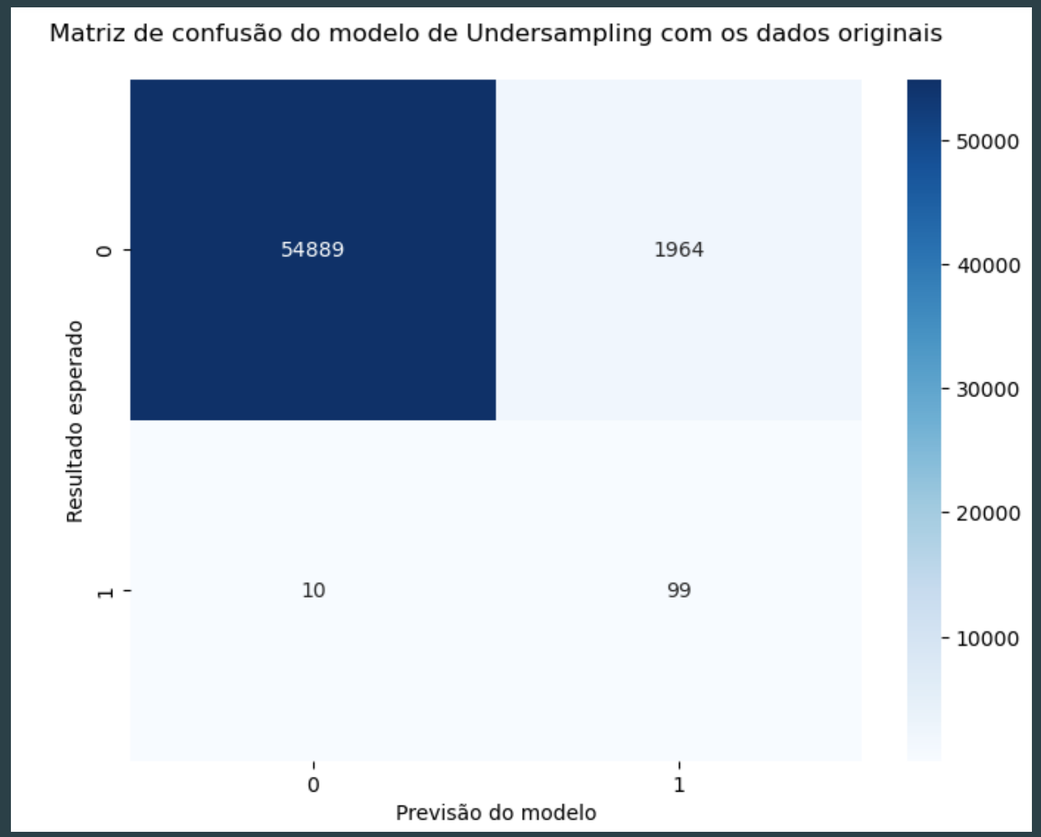
FRAUDE	PRECISION	RECALL	F1-SCORE
0	1	1	1
1	1	1	1

Acurácia: 1

Avaliação dos resultados

Undersampling – base original

Matrix de confusão



Métricas de classificação

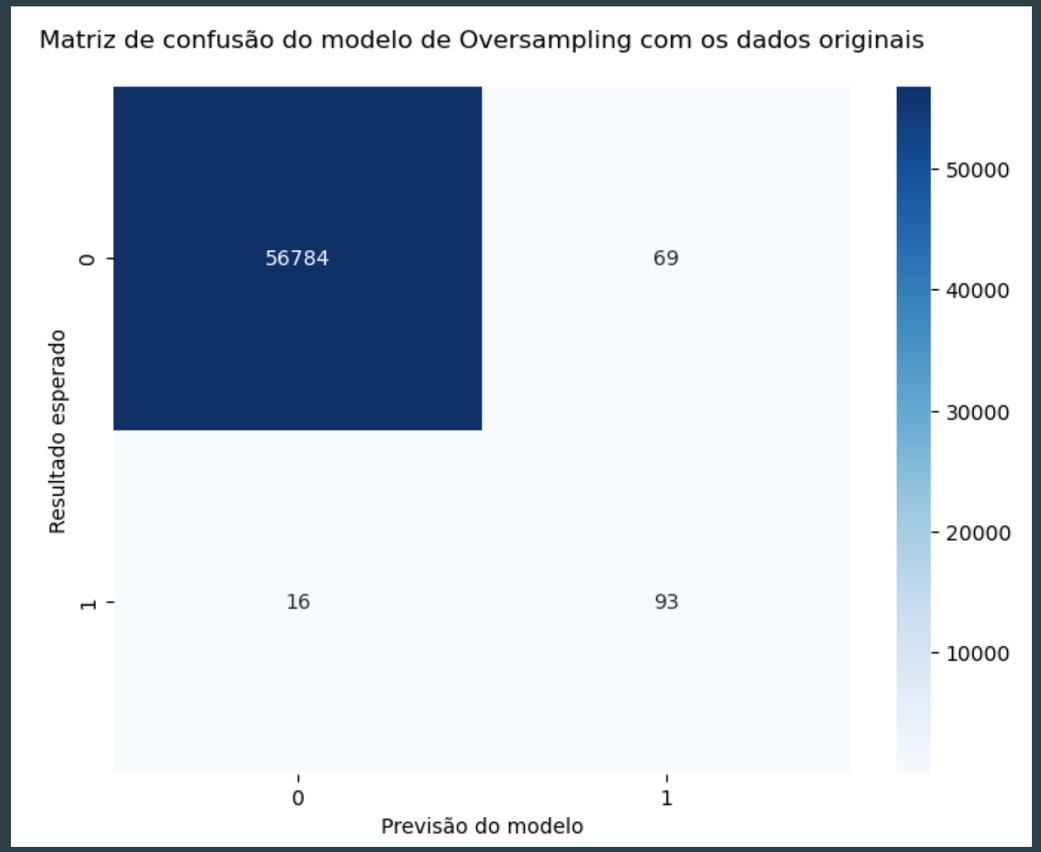
FRAUDE	PRECISION	RECALL	F1-SCORE
0	1	0.97	0.98
1	0.05	0.91	0.09

Acurácia: 0.97

Avaliação dos resultados

Oversampling – base original

Matrix de confusão



Métricas de classificação

FRAUDE	PRECISION	RECALL	F1-SCORE
0	1	1	1
1	0.57	0.85	0.69

Acurácia: 1

Conclusão

- O modelo de Oversampling é o que cumpre melhor com o objetivo proposto.
- Idealmente, os dados originais deveriam estar mais balanceados.