Credit Card Fraud Detection

Abstract

Credit card fraud, especially in the digital age, poses a great threat. Digital transactions are on the rise which in turn has increased the complexity in detecting fraud. This report aims to analyze applied supervised machine learning to the IEEE-CIS Fraud Detection dataset to classify transactions as fraudulent or legitimate. The main challenges were the class imbalance (fraudulent transactions are less than 5% of the data), high dimensionality (434 features), and a great deal of missing data. The analysis involved exploratory data analysis, specific data preprocessing. Three models used are Logistic Regression, Random Forest, and LightGBM. These models were trained and evaluated using the Area Under the Receiver Operating Characteristic Curve (AUC) as the main performance metric. LightGBM was shown to be the best of the models with a test AUC of 0.9442 which is higher than Random Forest (0.9342) and Logistic Regression (0.8698). The analysis concludes that gradient-boosting ensembles like LightGBM are well-suited for handling the complexity and imbalance seen in large-scale financial data and fraud detection tasks.

Introduction

Credit card fraud is a global problem which emphasizes the need for quick, efficient, and accurate data mining solutions. The challenge in developing models that can identify rare, malicious transactions, is that they are hidden amongst a large number of real transactions. This assignment tests the performance of different machine learning models on real world Fraud Detection data. The main goal is to test if advanced, tree-based ensembles can outperform a classical linear model.

Problem Statement and Data Sources

The analysis is based on the IEEE-CIS Fraud Detection dataset found on Kaggle (link can be found in the Bibliography). For this assignment, only the training dataset was used since its size (over 590,000 rows with 434 columns) was enough to model off of without putting too much computational strain on my system. The features include transaction metadata, identity information, and a large number of hidden variables (V1–V339). The binary target variable, isFraud, indicates whether a transaction was fraudulent (1) or legitimate (0).

Exploratory Data Analysis (EDA)

The data we are working with is very large and contains hundreds of features which makes exploratory analysis critical in understanding what is going on in the dataset and how to proceed with pre-processing.

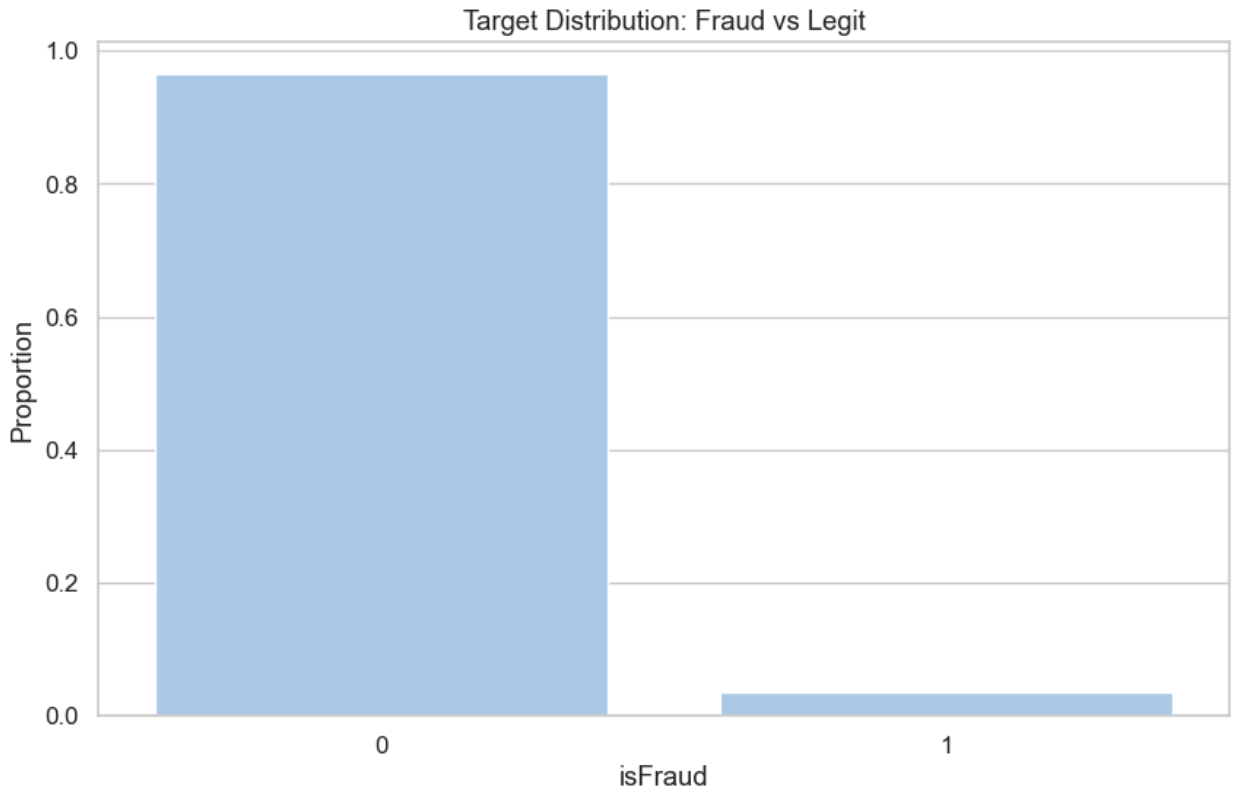


Figure 1: Target variable distribution

Looking at the distribution of fraudulent cases vs legitimate transactions (Figure 1), we can see there's a very small percentage of fraudulent charges. This aligns with real world cases since fraud does not occur as often.

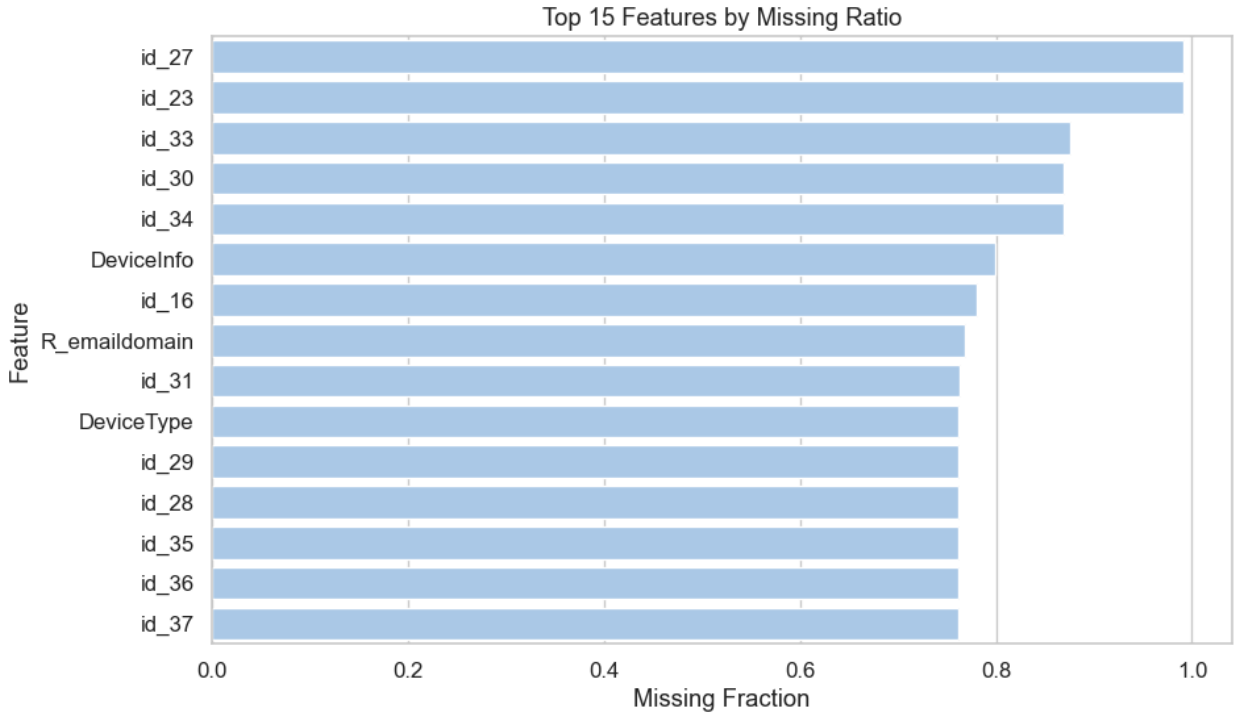
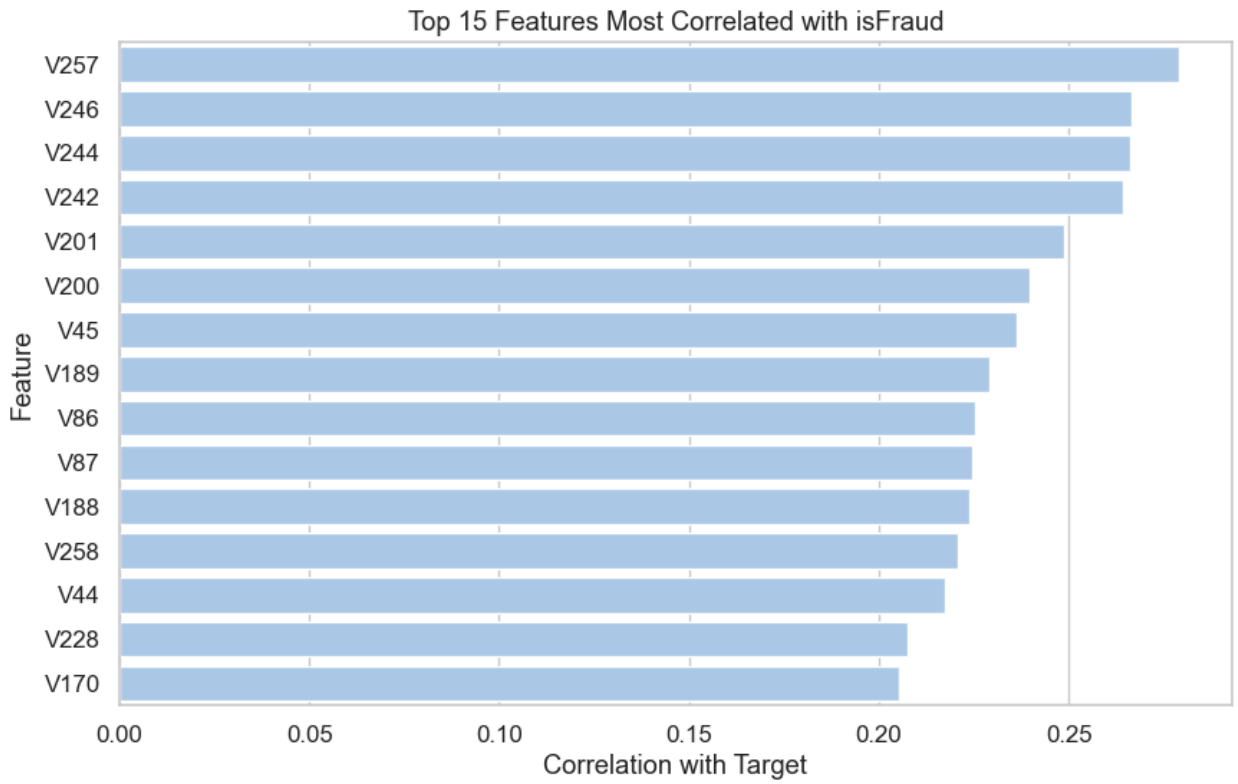


Figure 2: Top 15 features with the highest proportion of missing values

Exploring the data revealed that many columns had a great deal of missing information. This is not out of the ordinary as not all bank transactions are the same and involve the same level of detail. Many features, such as id_27, id_23, and DeviceInfo, have extreme levels of missing data, exceeding 80% (Figure 2) which will need to be handled before running any models.



Several of the hidden features (V257, V246, V244) showed a strong correlation with the target variable (Figure 3). Given there were many columns with missing data, this suggests that there are still many features with predictive importance that can be used in our models.

Methodology

Pre-processing

To address memory constraints and issues found in the EDA, a sample of the data was used for model training. Features with over 80% missing values were left as in only for models like LightGBM that can handle missing data. For other models, numerical features were imputed with the median, and categorical variables were encoded using label encoding.

Model Selection

- **Logistic Regression**: A linear baseline model.
- **Random Forest**: A bagging-based ensemble method.
- **LightGBM**: A gradient-boosting framework known for its speed and efficiency with large datasets.

Model Evaluation

Each models' performance was evaluated using the Area Under the Curve (AUC) metric, which is best suited to this data because of the previously seen class imbalance. A 3-fold stratified cross-validation was performed on the sampled data to test model stability and generalizability. Final models were also evaluated on a testing subset of the data.

Results- Model Performance

The models were compared using cross-validation and a test set of the data.

Cross-Validation Results

Model Name	Mean AUC
Logistic Regression (Base model)	0.8652 ± 0.0070
Random Forest	0.8997 ± 0.0068
Light GBM (Boosting model)	0.9259 ± 0.0070

Table 1: Mean AUC results from Cross-Validation on each model

The cross validation results show that the LightGBM model had the best performance with an AUC of 0.9259. This is closely followed by the AUC of Random Forest model which is 0.8997 and Logistic Regression which is 0.8652.

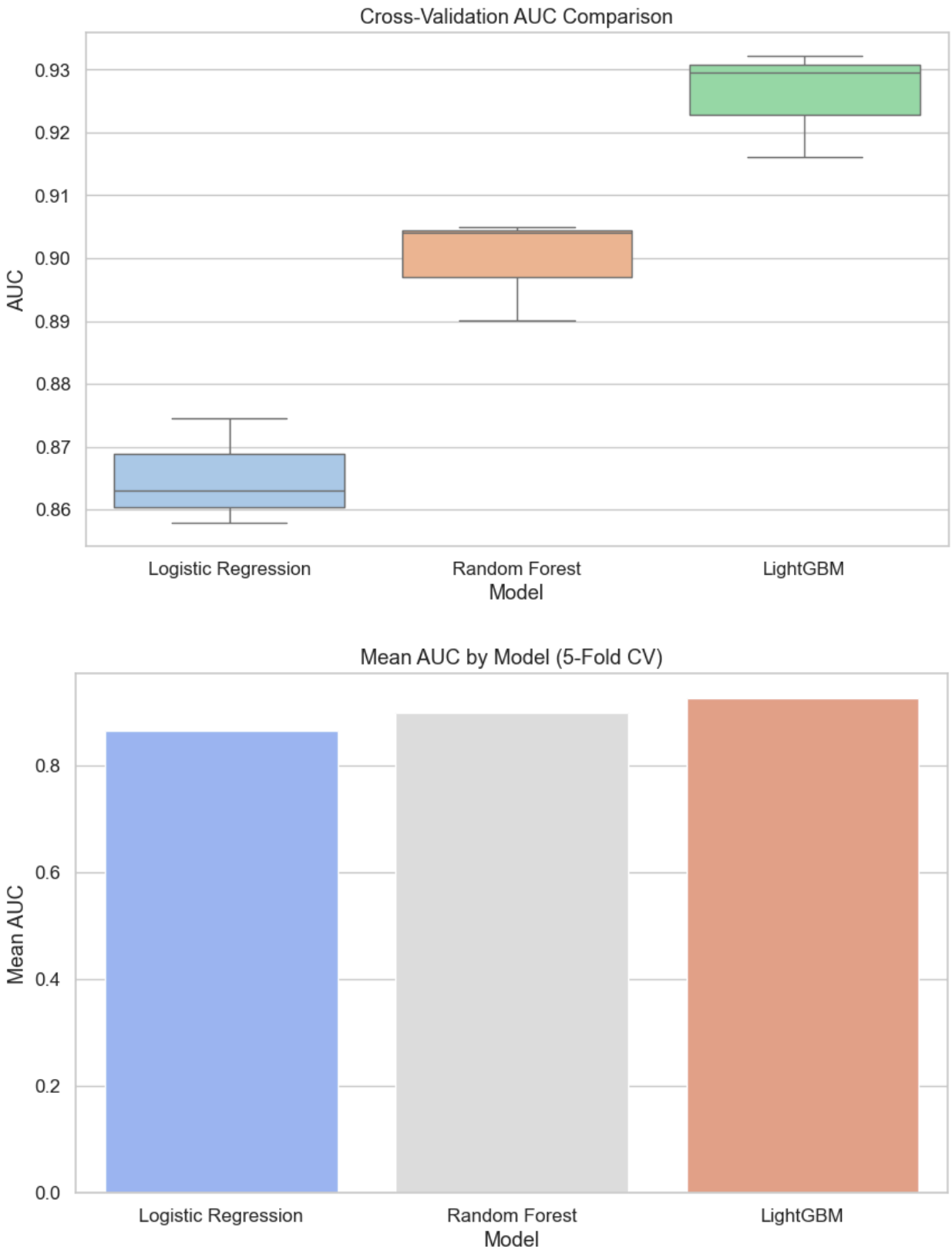


Figure 4:Cross-validation AUC comparison across the three models

Testing Set Results

Model Name	AUC
Logistic Regression (Base model)	0.8698
Random Forest	0.9342
Light GBM (Boosting model)	0.9442

Table 1: AUC results from Test data

When evaluated on the test set, the performance was similar to cross-validation. LightGBM delivered the best performance (AUC = 0.9442) with Random forest close behind with an AUC of 0.9342. Logistic regression, again, had the lowest performance.

Conclusions

The results of the analysis show that ensemble methods, particularly gradient-boosting, are superior to a linear model for this high-dimensional dataset. LightGBM's ability to handle missing data and complex interactions is what led to its performance. Random Forest also performed fairly well, while Logistic Regression, though simple, was not able to capture as much of the complexity of the data, leading to underfitting.

Limitations and Future Work

This analysis was done under some computational constraints that required data sampling and a limit to the number of cross-validation folds. Future work could incorporate cloud computing to train models without the worry of available memory. Also, the hidden names of many features made it difficult to interpret the results. However, this was due to the data being sourced from a past competition with privacy limits.

Lessons Learned

- Computational Cost: Given the dataset's size and high dimensionality, full-scale training was hard due to the memory strain it created. Instead, data sampling and using models like LightGBM that have good performance while keeping reasonable run times and memory use.
- Data Preprocessing: The quality of data is as important as the model choice. Proper handling of missing values and categorical features is both difficult and very necessary for precise model training and testing. It can be difficult to balance the need for clean data and the risk of losing value by over-handling missing information

Appendix

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score

import lightgbm as lgb
import matplotlib.pyplot as plt
import seaborn as sns



### Data
train_df = pd.read_csv("train_transaction.csv")
id_df = pd.read_csv("train_identity.csv")
data = train_df.merge(id_df, on="TransactionID", how="left")

X = data.drop(columns=["isFraud"])
y = data["isFraud"]
```

```
## split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=7406
)

## Missing data
#
num_cols = X_train.select_dtypes(include=[np.number]).columns
imputer = SimpleImputer(strategy="median")
X_train[num_cols] = imputer.fit_transform(X_train[num_cols])
X_test[num_cols] = imputer.transform(X_test[num_cols])

# scale
scaler = StandardScaler()
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])

### EDA
sns.set(style="whitegrid", palette="pastel", font_scale=1.1)
plt.rcParams["figure.figsize"] = (10, 6)

# fraud vs legit
fraud_counts = y_train.value_counts(normalize=True)
sns.barplot(x=fraud_counts.index, y=fraud_counts.values)
plt.title("Target Distribution: Fraud vs Legit")
```

```
plt.xlabel("isFraud")

plt.ylabel("Proportion")

plt.show()


# missing data

missing_ratio = X_train.isnull().mean().sort_values(ascending=False)

top_missing = missing_ratio.head(15)

sns.barplot(x=top_missing.values, y=top_missing.index)

plt.title("Top 15 Missing Features")

plt.xlabel("Missing Fraction")

plt.ylabel("Feature")

plt.show()


# corr graph

num_cols = X_train.select_dtypes(include=np.number).columns

corr = X_train[num_cols].corrwith(y_train).dropna().sort_values(key=abs,
ascending=False)[:15]

sns.barplot(x=corr.values, y=corr.index)

plt.title("Top 15 Features Most Correlated with isFraud")

plt.xlabel("Correlation with Target")

plt.ylabel("Feature")

plt.show()



### Features

cat_cols = X_train.select_dtypes(include=["object"]).columns
```

```
num_cols = X_train.select_dtypes(exclude=["object"]).columns

# one-hot encode
X_train_enc = pd.get_dummies(X_train, columns=cat_cols, drop_first=True)
X_test_enc  = pd.get_dummies(X_test, columns=cat_cols, drop_first=True)

X_test_enc = X_test_enc.reindex(columns=X_train_enc.columns, fill_value=0)

### Logistic Regression
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train_enc, y_train)

pred_lr = logreg.predict_proba(X_test_enc)[:, 1]
auc_lr = roc_auc_score(y_test, pred_lr)
print(f"Logistic Regression AUC: {auc_lr:.4f}")


### Random Forest
rf = RandomForestClassifier(n_estimators=200, random_state=7406, n_jobs=-1)
rf.fit(X_train_enc, y_train)
pred_rf = rf.predict_proba(X_test_enc)[:, 1]

auc_rf = roc_auc_score(y_test, pred_rf)
print(f"Random Forest AUC: {auc_rf:.4f}")
```

```
### Clean columns for gbm
# column names
X_train_enc.columns = X_train_enc.columns.str.replace('[^A-Za-z0-9_]+', '_', regex=True)
X_test_enc.columns  = X_test_enc.columns.str.replace('[^A-Za-z0-9_]+', '_', regex=True)

# remove duplicates
X_train_enc = X_train_enc.loc[:, ~X_train_enc.columns.duplicated()]
X_test_enc  = X_test_enc.loc[:, ~X_test_enc.columns.duplicated()]

X_test_enc = X_test_enc.reindex(columns=X_train_enc.columns, fill_value=0)

### Light GBM
lgb_train = lgb.Dataset(X_train_enc, y_train)
lgb_eval  = lgb.Dataset(X_test_enc, y_test, reference=lgb_train)

params = {
    "objective": "binary",
    "metric": "auc",
    "boosting_type": "gbdt",
    "learning_rate": 0.05,
    "num_leaves": 64,
    "verbose": -1
}
```

```
gbm = lgb.train(
    params,
    lgb_train,
    valid_sets=[lgb_eval],
    num_boost_round=200,
    callbacks=[lgb.early_stopping(stopping_rounds=10)]
)

pred_lgb = gbm.predict(X_test_enc)
auc_lgb = roc_auc_score(y_test, pred_lgb)
print(f"LightGBM AUC: {auc_lgb:.4f}")



### Cross-Validation
#cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=7406)

models = {
    "Logistic Regression": LogisticRegression(max_iter=1000, n_jobs=-1),
    "Random Forest": RandomForestClassifier(n_estimators=200, n_jobs=-1,
random_state=7406),
    "LightGBM": lgb.LGBMClassifier(
        n_estimators=200,
        learning_rate=0.05,
        num_leaves=64,
        subsample=0.8,
        colsample_bytree=0.8,
```

```
        random_state=7406
    )
}

cv_results = {}

cv = StratifiedKFold(n_splits=3, shuffle=True, random_state=7406)

for name, model in models.items():
    print(f"\n{name} Cross-Validation (3-fold, sampled):")
    X_sample = X_train_enc.sample(frac=0.25, random_state=42)
    y_sample = y_train.loc[X_sample.index]
    aucs = cross_val_score(model, X_sample, y_sample, cv=cv, scoring="roc_auc", n_jobs=-1)
    cv_results[name] = aucs
    print(f"AUCs: {np.round(aucs, 4)}")
    print(f"Mean AUC: {np.mean(aucs):.4f} ± {np.std(aucs):.4f}")



### Summary graphs
cv_df = (
    pd.DataFrame(cv_results)
    .melt(var_name="Model", value_name="AUC")
)

sns.boxplot(data=cv_df, x="Model", y="AUC", palette="pastel")
```

```
plt.title("Cross-Validation AUC Comparison")
plt.xlabel("Model")
plt.ylabel("AUC")
plt.show()


mean_aucs = {k: np.mean(v) for k, v in cv_results.items()}
sns.barplot(x=list(mean_aucs.keys()), y=list(mean_aucs.values()), palette="coolwarm")
plt.title("Mean AUC by Model (5-Fold CV)")
plt.xlabel("Model")
plt.ylabel("Mean AUC")
plt.show()
```

Bibliography and Credits

IEEE-CIS Fraud Detection Dataset (Kaggle, 2019):
https://www.kaggle.com/c/ieee-fraud-detection