FindDefault (Prediction of Credit Card fraud)

===

1. Report (PDF)

Description of Design Choices:

- **Data Understanding and Preprocessing:**
  - Describe the dataset, its size, features, and the presence of missing values or outliers.
  - Explain imputation techniques (if any) used to handle missing values.
  - Explain methods used to address outliers (e.g., IQR-based clipping).
- **Imbalanced Dataset Handling:**
  - Highlight the class imbalance issue.
  - Justify the choice of under-sampling to create a balanced dataset.
- **Feature Engineering and Selection:**
  - Explain the correlation analysis for feature selection.
  - Discuss the potential benefits of additional feature engineering techniques (e.g., PCA, feature scaling).
- **Model Selection and Evaluation:**
  - List the models explored (Logistic Regression, SVM, KNN, Random Forest, Gradient Boosting, XGBoost).
  - Explain the reasoning behind choosing hyperparameter tuning with GridSearchCV.
  - Justify the choice of evaluation metrics (accuracy, precision, recall, F1-score, ROC-AUC score) and why they are important in the context of fraud detection.

Performance Evaluation:

- Present the classification reports and confusion matrices for both the training and testing sets.
- Provide a table comparing the key performance metrics across the train and test sets.
- Interpret the results, highlighting:
  - The best performing model and its hyperparameters.
  - The model's strengths in detecting fraudulent transactions (high recall).
  - The trade-off between precision and recall, considering the cost of false positives and false negatives.

Discussion of Future Work:

- **Over-sampling and SMOTE:** Discuss these alternative techniques to handle imbalanced data.
- **Advanced Feature Engineering:** Explore dimensionality reduction (PCA), feature scaling, and the creation of new features for potentially improving model performance.
- **Cost-Sensitive Learning:** Investigate assigning different costs to false positives and false negatives to reflect the real-world impact of those errors.
- **Ensemble Techniques:** Explore methods like bagging, boosting, and stacking to potentially improve model robustness and accuracy.
- **Time Series Analysis:** Consider incorporating time-related features to capture trends and patterns in fraudulent behavior.

2. Source Code (Python)

The provided Python code demonstrates a structured and effective approach. Here's a breakdown of the key areas to emphasize:

Data Preparation

- Clear and concise comments explaining each step.
- Handling missing values (though not present in the provided dataset).
- Using IQR for outlier treatment.

Feature Selection

- **Correlation Analysis:** Clearly explained correlation-based feature selection, justifying the choice of thresholds (0.2).

Balancing Data

- Well-explained code demonstrating under-sampling.

Model Training and Evaluation

- **Hyperparameter Tuning:** Efficient implementation using GridSearchCV.
- **Diverse Model Selection:** Demonstrates exploration of various classification algorithms.
- **Comprehensive Evaluation:** Includes accuracy, precision, recall, F1-score, and ROC-AUC score in reporting.

	Accuracy	Precision	Recall	F1 Score	Roc Auc Score
Train	0.997354	1.000000	0.994709	0.997347	0.997354
Test	0.921053	0.987805	0.852632	0.915254	0.921053

3. Zip File Contents

- `report.pdf`: Contains the report outlined above.
- `creditcard_fraud_detection.ipynb` (or similarly named): The main Jupyter Notebook Python code file, well-commented and structured.
- `creditcard_fraud_detection.pdf` (or similarly named): The PDF format of Jupyter Notebook Python code file, well-commented and structured.

Production Folder Contents:

- `app.py`:
  - Serves as the main Python script for the UI-based ML model interaction.
  - Includes a clear and well-commented code structure for easy navigation and understanding.
  - Utilizes libraries such as Streamlit and scikit-learn to create a user-friendly interface.

- `iqr_thresholds.json`: Stores the IQR thresholds used for outlier treatment.
- `clf.pkl`: The saved, serialized model ready for deployment.

Additional Considerations

- **Model Deployment Strategy:** For deploying the model in a production setting, I used a streamlit python library. This will allow for easy integration of the API and loading of the final model into the application.