AI Credit Scoring System Specification

Overview

This document describes the specifications of the AI Credit Scoring System.
It covers AI training (train_model.py), web interface (webapp.py),
input data (input.csv), model behavior, evaluation metrics, and interpretation methods.

1. File Structure

File	Description
train_model.py	Trains the LightGBM model and exports evaluation files
webapp.py	Flask app for score visualization with risk highlighting
input.csv	Pre-processed data used for scoring
model.pkl	Serialized trained model
cat_maps.pkl	Encoded category mappings for categorical features
individual_scores.csv	Inference results including Name and score

2. Input Data (input.csv)

2.1 Columns

Column	Required	Type	Description
Name	Yes	int	Unique 1-based ID for each individual
Sex	Yes	str	"Man" or "Woman"
Marital	Yes	str	"Single" or "Married"
Age	Yes	int	Age (19–79 valid)
Income	Yes	float	Annual income > 0
CreditAppAmount	Yes	float	Amount applied for credit > 0
OtherDebts	Yes	float	Existing other debts >= 0
DebtRestruct	Yes	int	0 or 1; indicates history of restructuring
EmploymentYears	Yes	float	Years employed (>= 0, <= Age–15)
Occupation	Optional	str	Category of occupation
Industry	Optional	str	Industry name
Education	Optional	str	Education level
Dependents	Optional	int	Number of dependents
OwnHouse	Optional	int	0 or 1; ownership of house
Foreigner	Optional	int	0 or 1; foreign national flag
Phone	Optional	int	0 or 1; phone possession
Guarantor	Optional	int	0 or 1; has guarantor
Collateral	Optional	int	0 or 1; has collateral
BorrowingRatio	Calculated	float	OtherDebts / Income if Income > 0, else 0

3. Preprocessing

Strip whitespace and control characters from column headers
Replace empty strings with NaN
Validate required columns
Drop rows with missing required fields
Validate category values:
Sex: must be "Man" or "Woman"
Marital: must be "Single" or "Married"
Validate numeric ranges (e.g., Age > 18 and < 80)
Encode categorical columns using cat_maps.pkl

4. Model Training (train_model.py)

4.1 Model Type

LightGBM (LGBMClassifier)
Binary classification (Delinquency = 1 or 0)

4.2 Model Training (train_model.py)

1. Loads and cleans credit_data_learn.csv
2. Validates required fields and value ranges
3. Calculates BorrowingRatio (OtherDebts / Income)
4. Encodes categorical variables (e.g., Sex, Marital)
5. Saves category mappings to cat_maps.pkl
6. Splits data into training and test sets
7. Trains LightGBM model with custom hyperparameters
8. Computes SHAP values for test data
9. Under overall model evaluation, Accuracy and AUC are displayed to represent prediction correctness and class separability.
10. The classification report is displayed, including precision, recall, and f1-score for each class.
11. Outputs individual_scores.csv with Name, prediction, and actual
12. Saves input.csv (used in WebApp) and model.pkl

4.3 Hyperparameters

Parameter	Value	Description
n_estimators	700	Number of boosting iterations (trees). Higher values may improve accuracy but increase training time.
learning_rate	0.0055	Step size shrinkage used in update to prevent overfitting. Smaller values lead to slower, more stable learning.
max_depth	4	Maximum depth of each tree. Shallower trees generalize better.
min_child_samples	8	Minimum number of data samples required in a leaf node. Controls complexity.
subsample	1	Fraction of data used in each iteration. 1.0 means all data is used.
colsample_bytree	0.7	Fraction of features used per tree. Adds randomness and helps reduce overfitting.
reg_alpha	1	L1 regularization term. Encourages sparsity (zeroing out unimportant features).
reg_lambda	1	L2 regularization term. Helps prevent overfitting by penalizing large weights.
feature_fraction	0.8	Fraction of features to use in each iteration. Helps reduce overfitting.
class_weight	{0: 1.0, 1: 6}	Assigns weights to each class. {0: 1.0, 1: 6.0} means delinquent class is 6× more important.

4.4 Output

Object	Description
model.pkl	Trained model
cat_maps.pkl	category encoding mappings
individual_scores.csv	inference result
input.csv	processed input file

5. Web-based Scoring (webapp.py)
1. Reads input.csv and model.pkl
2. Predicts delinquency probability (0.0000–1.0000)
3. Renders HTML table with .risk-high class if score >= 0.7
4. Reverse maps category codes to labels
5. Displays Name, demographic features, and Score

6. Evaluation Metrics

Metric	Description
Accuracy	Correct predictions / Total predictions
AUC	Area under ROC curve; closer to 1 = better
Precision	TP / (TP + FP); higher = fewer false positives
Recall	TP / (TP + FN); higher = fewer false negatives
Classification Report	Includes precision, recall, f1-score per class

7. SHAP Interpretation
SHAP (SHapley Additive exPlanations) values are extracted per individual
For each test individual, top 3 influencing features are shown
Each feature is labeled:
Positive (increased probability)
Negative (decreased probability)
Helpful for transparency and compliance

8. Notes
Name is treated as 1-based integer (e.g., 1 to 1000)
Scores are based on test split (not full data)
SHAP values apply only to test set individuals
All outputs are UTF-8 with BOM for compatibility (e.g., Excel)

9. Licensing
Current license: TBD (suggested: MIT or Apache 2.0)
Model may be exported with restrictions depending on usage scope

10. How to Use the System
1. Training the AI Model
1.1. Edit credit_data_learn.csv
The following columns are mandatory and must be retained:
Name, Sex, Age, Marital, Income, CreditAppAmount, OtherDebts, DelinquencyInfo, DebtRestruct, Employment
Other columns can be modified or updated as needed.
2. Run the training script
```
$ python train_model.py
```
This will create the trained model (model.pkl) and category mapping file (cat_maps.pkl) in the models direct
It will also generate a formatted input.csv in the data directory for later use.

2. Using a Trained Model for Credit Evaluation
2.1. Edit input.csv
Update it with the credit evaluation data you want to score.
Ensure the format matches the one generated during training.
2.2. Run the web application
```
$ python webapp.py
```

3. Access the application in your browser
URL: http://localhost:8080/
The evaluation results will be displayed in a table.

Notes
Make sure all required Python packages are installed as described in Environment Setup.pdf.
If using a different port for Flask, update the port parameter in webapp.py.
The process flow is illustrated in the provided flow diagram.

Latest Updates & Resources
The latest updates, revision history, and general-purpose tools are available at the following URL:
https://github.com/NextGenAI-corder/FastSpring