**Credit Risk Predictor - Documentation**

**Table of Contents**

* [Introduction](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#introduction)
* [Application URL](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#application-url)
* [Repository](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#repository)
* [System Requirements](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#system-requirements)
* [Installation Guide](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#installation-guide)
* [Project Architecture](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#project-architecture)
* [Code Documentation](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#code-documentation)
  + [Data Preprocessing (creditrisk.py)](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#data-preprocessing-creditriskpy)
  + [Model Training (train\_models.py)](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#model-training-train_modelspy)
  + [Model Interpretation (shap\_analysis.py)](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#model-interpretation-shap_analysispy)
  + [Performance Visualization (performance\_plots.py)](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#performance-visualization-performance_plotspy)
  + [Web Application (credit\_risk\_streamlitapp.py)](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#web-application-credit_risk_streamlitapppy)
* [User Guide](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#user-guide)
* [API Reference](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#api-reference)
* [Model Performance](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#model-performance)
* [Feature Importance](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#feature-importance)
* [Troubleshooting](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#troubleshooting)
* [Roadmap](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#roadmap)
* [Contributing Guidelines](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#contributing-guidelines)
* [License Information](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#license-information)
* [Acknowledgments](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#acknowledgments)
* [Contact](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#contact)

**Introduction**

The Credit Risk Predictor is a machine learning-based application that assesses credit risk using German credit data. It implements a comprehensive ML pipeline including preprocessing, feature engineering, model training with ensemble techniques, and result visualization through an interactive web interface built with Streamlit.

This solution helps financial institutions make data-driven decisions on loan applications by providing risk assessments with interpretable justifications through SHAP analysis.

**Application URL**

The application is deployed and accessible at: <https://creditriskpredictor.streamlit.app/>

**Repository**

The source code is available on GitHub: <https://github.com/Kpreya/Credit_Risk_Predictor_App>

**System Requirements**

* **Python**: 3.8+ ([Download Python](https://www.python.org/downloads/))
* **Operating System**: Windows 10+, macOS 10.14+, or Linux
* **RAM**: Minimum 4GB (8GB recommended)
* **Disk Space**: 500MB+ (for application and dependencies)
* **Internet Connection**: Required for initial dependency installation and data access

**Installation Guide**

**Prerequisites**

* Python 3.8+ installed
* Git installed ([Download Git](https://git-scm.com/downloads))
* pip package manager (included with Python)

**Step 1: Clone the Repository**

git clone https://github.com/Kpreya/Credit\_Risk\_Predictor\_App.git

cd Credit\_Risk\_Predictor\_App

**Step 2: Create a Virtual Environment (Optional but Recommended)**

# For Windows

python -m venv venv

venv\Scripts\activate

# For macOS/Linux

python3 -m venv venv

source venv/bin/activate

**Step 3: Install Dependencies**

pip install -r requirements.txt

**Step 4: Download Dataset**

The German credit dataset should be placed in the data/ directory. If not included in the repository:

# Create data directory if it doesn't exist

mkdir -p data

# Download dataset (replace with actual data source if needed)

# Example using wget:

# wget -O data/german\_credit\_data.csv https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data

**Step 5: Verify Installation**

python -c "import pandas, numpy, sklearn, streamlit, xgboost, shap; print('All dependencies installed successfully')"

**Project Architecture**

Credit\_Risk\_Predictor\_App/

│

├── creditrisk.py # Data preprocessing and feature engineering

├── train\_models.py # Model training script

├── shap\_analysis.py # Model interpretation

├── performance\_plots.py # Performance visualization

├── credit\_risk\_streamlitapp.py # Web application

│

├── data/ # Data directory

│ ├── german\_credit\_data.csv # Original dataset

│ └── preprocessed\_credit\_data.csv # Processed dataset

│

├── models/ # Saved models and artifacts

│ ├── preprocessor.joblib # Feature preprocessing pipeline

│ ├── target\_encoder.joblib # Label encoder for target variable

│ ├── RandomForest\_model.joblib # Trained Random Forest model

│ ├── LogisticRegression\_model.joblib # Trained Logistic Regression model

│ ├── SVC\_model.joblib # Trained SVM model

│ ├── XGBoost\_model.joblib # Trained XGBoost model

│ ├── stacking\_ensemble.joblib # Trained stacking ensemble model

│ └── feature\_names.joblib # Saved feature names for interpretation

│

├── plots/ # Generated visualizations

│ ├── stacking\_ensemble\_cm.png # Confusion matrix

│ ├── stacking\_ensemble\_roc.png # ROC curve

│ ├── stacking\_ensemble\_shap\_summary\_bar.png # SHAP bar plot

│ └── stacking\_ensemble\_shap\_summary.png # SHAP summary plot

│

├── requirements.txt # Dependencies

├── README.md # Project overview

└── LICENSE # License information

**Data Flow**

1. Raw data is loaded and preprocessed by creditrisk.py
2. Preprocessed data is used for model training in train\_models.py
3. Trained models are analyzed in shap\_analysis.py and performance\_plots.py
4. The Streamlit app loads all artifacts for the web interface

**Code Documentation**

**Data Preprocessing (creditrisk.py)**

**Overview**

This script handles data loading, cleaning, preprocessing, feature engineering, train-test splitting, and artifact generation for downstream modeling.

**Key Components**

**Data Loading and Cleaning**

df = pd.read\_csv(DATA\_PATH)

df.columns = df.columns.str.strip().str.lower().str.replace(' ', '\_')

**Missing Value Imputation**

cat\_imp = SimpleImputer(strategy='most\_frequent')

df[cat\_cols] = pd.DataFrame(cat\_imp.fit\_transform(df[cat\_cols]), columns=cat\_cols)

num\_imp = SimpleImputer(strategy='median')

df[num\_cols] = pd.DataFrame(num\_imp.fit\_transform(df[num\_cols]), columns=num\_cols)

**Outlier Handling**

def cap\_outliers(df, cols):

df\_capped = df.copy()

for col in cols:

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 \* IQR

upper = Q3 + 1.5 \* IQR

df\_capped[col] = df\_capped[col].clip(lower, upper)

return df\_capped

df = cap\_outliers(df, num\_cols)

**Feature Engineering**

df['credit\_per\_month'] = df['credit\_amount'] / (df['duration'] + 1e-3)

df['age\_group'] = pd.cut(df['age'], bins=[18, 30, 40, 50, 60, 100],

labels=['18-30', '31-40', '41-50', '51-60', '60+'])

df['credit\_to\_age\_ratio'] = df['credit\_amount'] / df['age']

df['financial\_security'] = df['saving\_level'] / (df['credit\_amount'] / 1000 + 1)

**Train/Test Split with SMOTE**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, stratify=y, random\_state=42

)

smote = SMOTE(random\_state=42)

X\_train\_smote, y\_train\_smote = smote.fit\_resample(X\_train\_processed, y\_train)

**Outputs**

* Preprocessed dataset saved to data/preprocessed\_credit\_data.csv
* Various preprocessing artifacts saved to models/ directory

**Model Training (train\_models.py)**

**Overview**

This script trains multiple machine learning models on the preprocessed data and saves them for later use.

**Key Components**

**Model Definitions**

models = {

'RandomForest': RandomForestClassifier(n\_estimators=200, random\_state=42, n\_jobs=-1),

'LogisticRegression': LogisticRegression(max\_iter=500, solver='liblinear', random\_state=42),

'SVC': SVC(kernel='rbf', probability=True, random\_state=42),

'XGBoost': XGBClassifier(n\_estimators=200, learning\_rate=0.05, max\_depth=5,

subsample=0.8, colsample\_bytree=0.8, random\_state=42, n\_jobs=-1)

}

**Model Training Loop**

for name, model in models.items():

print(f"Training {name}...")

model.fit(X\_sampled, y\_sampled)

joblib.dump(model, os.path.join(MODEL\_PATH, f"{name}\_model.joblib"))

**Cross-Validation**

for name, model in models.items():

scores = cross\_val\_score(model, X\_sampled, y\_sampled, cv=5, scoring='accuracy')

print(f"{name}: Mean Accuracy = {scores.mean():.4f}, Std = {scores.std():.4f}")

**Outputs**

* Trained models saved to models/ directory

**Model Interpretation (shap\_analysis.py)**

**Overview**

This script uses SHAP (SHapley Additive exPlanations) to interpret the trained stacking ensemble model.

**Key Components**

**SHAP Explainer Setup**

def predict\_proba\_positive(data\_matrix):

df = pd.DataFrame(data\_matrix, columns=feature\_names)

return model.predict\_proba(df)[:, 1]

explainer = shap.KernelExplainer(predict\_proba\_positive, background)

**SHAP Value Computation**

shap\_values = explainer.shap\_values(X\_sample, nsamples=100)

**Visualization Generation**

shap.summary\_plot(shap\_values, X\_sample, plot\_type='bar', show=False)

plt.savefig(os.path.join(PLOT\_PATH, 'stacking\_ensemble\_shap\_summary\_bar.png'), dpi=300)

**Outputs**

* SHAP plots saved to plots/ directory

**Performance Visualization (performance\_plots.py)**

**Overview**

This script generates visualizations for model performance evaluation.

**Key Components**

**Confusion Matrix**

y\_pred = model.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred, labels=model.classes\_)

disp = ConfusionMatrixDisplay(cm, display\_labels=model.classes\_)

**ROC Curve**

y\_proba = model.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_proba)

roc\_disp = RocCurveDisplay(fpr=fpr, tpr=tpr)

**Outputs**

* Performance plots saved to plots/ directory

**Web Application (credit\_risk\_streamlitapp.py)**

**Overview**

This Streamlit application provides an interactive interface for credit risk prediction.

**Key Components**

**User Input Collection**

sex = st.sidebar.selectbox("Sex", df['sex'].unique())

age = st.sidebar.slider("Age", int(df['age'].min()), int(df['age'].max()), int(df['age'].median()))

# ... more inputs

**Feature Engineering on Input**

input\_df['credit\_per\_month'] = input\_df['credit\_amount'] / (input\_df['duration'] + 1e-3)

input\_df['age\_group'] = pd.cut(

input\_df['age'], bins=[18, 30, 40, 50, 60, 100],

labels=['18-30', '31-40', '41-50', '51-60', '60+']

)

# ... more feature engineering

**Prediction**

X\_proc = preprocessor.transform(input\_df)

probas = model.predict\_proba(X\_proc)[0]

proba\_bad = probas[0]

proba\_good = probas[1]

**Interactive Tabs**

tabs = st.tabs(["Prediction", "Exploration", "Performance", "Interpretation"])

**Running the App**

streamlit run credit\_risk\_streamlitapp.py

**User Guide**

**Starting the Application**

1. Follow the [Installation Guide](https://claude.ai/chat/0e05ee6b-ab0c-4841-8b07-380969fc624e#installation-guide) to set up the environment
2. Run the Streamlit application:
3. streamlit run credit\_risk\_streamlitapp.py
4. Open your web browser and navigate to [http://localhost:8501](http://localhost:8501/)

**Using the Application**

The application interface is divided into several tabs:

**1. Prediction Tab**

* Input client information using the sidebar controls
* View the predicted credit risk (good/bad) and associated probability
* Adjust the risk threshold slider to customize the decision boundary

**2. Exploration Tab**

* Select features from the dropdown to explore distributions
* Compare feature distributions between good and bad risk groups
* Analyze patterns in the data through interactive histograms

**3. Performance Tab**

* View the confusion matrix showing model prediction accuracy
* Examine the ROC curve displaying the true positive vs. false positive rate
* Understand model performance metrics

**4. Interpretation Tab**

* Explore feature importance through SHAP plots
* Understand which factors most influence credit risk predictions
* Gain insights into model decision-making process

**API Reference**

**Preprocessing Module**

# Load preprocessing pipeline

preprocessor = joblib.load(os.path.join(MODEL\_PATH, 'preprocessor.joblib'))

# Apply preprocessing to new data

X\_processed = preprocessor.transform(input\_df)

**Prediction Module**

# Load model

model = joblib.load(os.path.join(MODEL\_PATH, 'stacking\_ensemble.joblib'))

# Get predictions

prediction = model.predict(X\_processed)

probabilities = model.predict\_proba(X\_processed)

# Decode predictions

label\_encoder = joblib.load(os.path.join(MODEL\_PATH, 'target\_encoder.joblib'))

prediction\_label = label\_encoder.inverse\_transform(prediction)

**Feature Names**

# Load feature names for interpretation

feature\_names = joblib.load(os.path.join(MODEL\_PATH, 'feature\_names.joblib'))

**Model Performance**

The application uses a stacking ensemble combining multiple base classifiers:

* Random Forest
* Logistic Regression
* Support Vector Classifier (SVC)
* XGBoost

Performance metrics for the stacking ensemble:

* **Accuracy**: ~78% (based on test set)
* **AUC-ROC**: ~0.80
* **Precision/Recall Balance**: Optimized for credit risk use case

For detailed performance metrics and visualizations, refer to the Performance tab in the application.

**Feature Importance**

According to SHAP analysis, the most influential features for credit risk prediction are:

1. **Financial Security**: Ratio of savings level to credit amount
2. **Checking Account Status**: Level of funds in checking account
3. **Credit Amount**: Total loan amount requested
4. **Duration**: Loan term length
5. **Age**: Applicant's age
6. **Purpose**: Loan purpose category

The relative importance varies by individual application. For personalized analysis, see the Interpretation tab in the application.

**Troubleshooting**

**Common Issues and Solutions**

**Application Won't Start**

* **Issue**: ModuleNotFoundError when starting the application
* **Solution**: Verify all dependencies are installed with pip install -r requirements.txt

**Missing Files Error**

* **Issue**: FileNotFoundError when loading models or data
* **Solution**: Ensure you've run the preprocessing and model training scripts in sequence:
* python creditrisk.pypython train\_models.pypython shap\_analysis.pypython performance\_plots.py

**Memory Issues**

* **Issue**: Out of memory errors when training models
* **Solution**: Reduce sample size in train\_models.py by adjusting the resample parameter:
* X\_sampled, y\_sampled = resample(X\_train, y\_train, n\_samples=int(0.3 \* len(X\_train)), random\_state=42)

**Visualization Errors**

* **Issue**: Plots not displaying in the application
* **Solution**: Verify all plots exist in the plots/ directory. If missing, run:
* python performance\_plots.pypython shap\_analysis.py

**Roadmap**

**Planned Features**

* **API Endpoint**: REST API for programmatic risk assessment
* **Additional Models**: Deep learning approaches (LSTM, Transformer)
* **Custom Feature Engineering**: User-defined feature creation interface
* **Batch Processing**: Support for multiple applications assessment
* **Enhanced Visualizations**: Interactive 3D plots for feature relationships
* **Model Monitoring**: Drift detection and performance tracking
* **PDF Report Generation**: Downloadable credit assessment reports

**Timeline**

* Q2 2025: API Endpoint & Batch Processing
* Q3 2025: Enhanced Visualizations & PDF Reports
* Q4 2025: Model Monitoring & Additional Models

**Contributing Guidelines**

We welcome contributions to improve the Credit Risk Predictor! Please follow these steps:

1. **Fork the repository** on GitHub
2. **Create a feature branch**:
3. git checkout -b feature/your-feature-name
4. **Implement your changes** with clear code and comments
5. **Write tests** for new functionality
6. **Update documentation** to reflect your changes
7. **Submit a pull request** with a clear description of improvements

**Code Style**

* Follow [PEP 8](https://pep8.org/) guidelines
* Use meaningful variable and function names
* Include docstrings for functions and classes
* Maintain >80% test coverage

**Issue Reporting**

Please use the [GitHub issue tracker](https://github.com/Kpreya/Credit_Risk_Predictor_App/issues) to report bugs or request features.

**License Information**

This project is licensed under the MIT License - see the [LICENSE](https://claude.ai/chat/LICENSE) file for details.

**MIT License Summary**

* Permission to use, copy, modify, and distribute the software
* The software is provided "as is", without warranty of any kind
* The authors or copyright holders are not liable for any claims or damages

**Acknowledgments**

* [German Credit Dataset](https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)) from UCI Machine Learning Repository
* [Scikit-learn](https://scikit-learn.org/) for machine learning algorithms
* [XGBoost](https://xgboost.readthedocs.io/) for gradient boosting implementation
* [SHAP](https://github.com/slundberg/shap) for model interpretation
* [Streamlit](https://streamlit.io/) for web application framework
* [Plotly](https://plotly.com/) for interactive visualizations

**Contact**

**Developer**

* **Krishnopreya**
* GitHub: [Kpreya](https://github.com/Kpreya)

**Report Issues**

* Submit bug reports and feature requests on [GitHub Issues](https://github.com/Kpreya/Credit_Risk_Predictor_App/issues)

*Last updated: April 26, 2025*