Credit Scoring & Fraud Detection Model

**Stage 2: Advanced EDA & Feature Engineering**

Advanced Data Science & Feature Engineering Documentation  
Generated: August 03, 2025  
Version: 2.0

# Table of Contents

1. 1. Executive Summary
2. 2. Stage 2 Philosophy & Approach
3. 3. Advanced Exploratory Data Analysis
4. 4. Statistical Analysis Framework
5. 5. Feature Engineering Architecture
6. 6. Domain-Driven Feature Creation
7. 7. Data Visualization Strategy
8. 8. Feature Selection & Optimization
9. 9. Technical Implementation Details
10. 10. Quality Assurance & Validation
11. 11. Results & Key Insights
12. 12. Next Steps & Integration

# 1. Executive Summary

Stage 2 represents a comprehensive deep-dive into the data science foundation of our fraud detection system. This phase focused on extracting maximum insights from the credit card transaction dataset through advanced exploratory data analysis (EDA) and sophisticated feature engineering techniques.

The work completed in this stage transforms raw transaction data into a rich, information-dense feature space that captures complex patterns indicative of fraudulent behavior. Through statistical analysis, domain expertise, and advanced feature creation techniques, we have established a robust foundation for high-performance machine learning models.

## Stage 2 Key Achievements

* • Comprehensive statistical analysis revealing significant fraud patterns
* • Advanced feature engineering creating 100+ sophisticated features
* • Domain-driven feature creation based on fraud detection expertise
* • Interactive visualization suite for data exploration
* • Automated feature selection and optimization pipeline
* • Correlation analysis identifying key predictive relationships
* • Temporal pattern analysis revealing time-based fraud indicators
* • Risk scoring framework for composite fraud assessment

# 2. Stage 2 Philosophy & Approach

## 2.1 Data-Driven Discovery Philosophy

Our approach to EDA and feature engineering is grounded in the principle of 'Data-Driven Discovery' - letting the data reveal its patterns while applying domain expertise to guide the exploration. This philosophy ensures we capture both obvious and subtle indicators of fraudulent behavior.

## 2.2 Multi-Layered Analysis Strategy

The analysis follows a systematic, multi-layered approach:

* • Descriptive Layer: Understanding basic data characteristics and distributions
* • Statistical Layer: Applying rigorous statistical tests to identify significant patterns
* • Correlation Layer: Discovering relationships between features and fraud indicators
* • Temporal Layer: Analyzing time-based patterns and seasonality
* • Domain Layer: Incorporating fraud detection domain knowledge
* • Engineering Layer: Creating sophisticated derived features

## 2.3 Feature Engineering Principles

Feature engineering follows these core principles:

* • Domain Relevance: Every feature should have a logical connection to fraud detection
* • Statistical Significance: Features must demonstrate statistical relationship with fraud
* • Interpretability: Complex features maintain explainable business logic
* • Robustness: Features should be stable across different data distributions
* • Scalability: Feature creation process must handle large-scale data efficiently

# 3. Advanced Exploratory Data Analysis

## 3.1 Comprehensive Dataset Profiling

The EDA framework provides comprehensive dataset profiling including:

* • Dataset overview with transaction counts and fraud rates
* • Statistical distribution analysis for all numerical features
* • Class imbalance quantification and severity assessment
* • Missing value patterns and data quality assessment
* • Outlier detection and anomaly identification
* • Feature correlation analysis and multicollinearity detection

## 3.2 Statistical Hypothesis Testing

Advanced statistical testing framework to identify significant differences between fraudulent and normal transactions:

# Mann-Whitney U Test for each feature  
for feature in numerical\_features:  
 fraud\_values = fraud\_df[feature]  
 normal\_values = normal\_df[feature]  
   
 statistic, p\_value = stats.mannwhitneyu(  
 fraud\_values, normal\_values,   
 alternative='two-sided'  
 )  
   
 # Determine statistical significance  
 significant = p\_value < 0.05

## 3.3 Class Imbalance Analysis

Detailed analysis of class imbalance reveals critical insights for model training:

* • Quantification of imbalance ratio (typically 99.4% normal, 0.6% fraud)
* • Assessment of imbalance severity and impact on model performance
* • Identification of optimal resampling strategies
* • Analysis of fraud distribution across different feature segments

# 4. Statistical Analysis Framework

## 4.1 Distribution Analysis

Comprehensive analysis of feature distributions reveals important characteristics:

* • Normality Testing: Shapiro-Wilk tests to assess normal distribution assumptions
* • Skewness Analysis: Identifying asymmetric distributions requiring transformation
* • Kurtosis Evaluation: Detecting heavy-tailed distributions and outliers
* • Distribution Comparison: Comparing fraud vs. normal transaction distributions

## 4.2 Correlation Analysis

Multi-dimensional correlation analysis to understand feature relationships:

# Comprehensive correlation analysis  
corr\_matrix = df.corr()  
  
# Target correlations (fraud indicator)  
target\_correlations = corr\_matrix['Class'].abs().sort\_values(ascending=False)  
  
# High correlation pairs (potential multicollinearity)  
high\_corr\_pairs = []  
for i in range(len(corr\_matrix.columns)):  
 for j in range(i+1, len(corr\_matrix.columns)):  
 corr\_val = abs(corr\_matrix.iloc[i, j])  
 if corr\_val > 0.7:  
 high\_corr\_pairs.append({  
 'feature1': corr\_matrix.columns[i],  
 'feature2': corr\_matrix.columns[j],  
 'correlation': corr\_matrix.iloc[i, j]  
 })

## 4.3 Temporal Pattern Analysis

Time-based analysis reveals important fraud patterns:

* • Hourly fraud rate variations identifying peak risk periods
* • Day-of-week patterns showing fraud concentration
* • Seasonal trends in fraudulent activity
* • Transaction timing anomalies as fraud indicators

# 5. Feature Engineering Architecture

## 5.1 Systematic Feature Creation Framework

The feature engineering architecture follows a systematic approach to create comprehensive feature sets:

* • Temporal Features: Time-based patterns and cyclical encodings
* • Amount Features: Transaction value transformations and binning
* • PCA Interactions: Cross-products and combinations of PCA components
* • Statistical Features: Rolling statistics and aggregations
* • Domain Features: Fraud-specific risk indicators
* • Polynomial Features: Non-linear feature combinations
* • Risk Scores: Composite fraud risk assessments

## 5.2 Advanced Temporal Feature Engineering

Sophisticated temporal features capture time-based fraud patterns:

# Advanced temporal feature creation  
df['Hour'] = (df['Time'] / 3600) % 24  
df['Day'] = (df['Time'] / (3600 \* 24)) % 7  
  
# Cyclical encoding for temporal features  
df['Hour\_sin'] = np.sin(2 \* np.pi \* df['Hour'] / 24)  
df['Hour\_cos'] = np.cos(2 \* np.pi \* df['Hour'] / 24)  
  
# Risk period indicators  
df['Is\_late\_night'] = ((df['Hour'] >= 23) | (df['Hour'] <= 5)).astype(int)  
df['Is\_business\_hours'] = ((df['Hour'] >= 9) & (df['Hour'] <= 17)).astype(int)  
df['Is\_weekend'] = (df['Day'] >= 5).astype(int)

## 5.3 Amount-Based Feature Engineering

Transaction amount features capture value-based fraud patterns:

* • Logarithmic transformations to handle skewed distributions
* • Z-score normalization for outlier detection
* • Percentile-based ranking for relative positioning
* • Round number detection for suspicious patterns
* • Amount binning for categorical analysis
* • Decimal place analysis for precision patterns

# 6. Domain-Driven Feature Creation

## 6.1 Fraud Detection Domain Knowledge

Feature creation incorporates deep domain knowledge about fraud patterns:

* • High-Value Transactions: Large amounts may indicate money laundering attempts
* • Micro-Transactions: Very small amounts might be card testing behavior
* • Round Amounts: Suspicious round numbers (e.g., exactly $100, $500)
* • Unusual Timing: Transactions during odd hours (3-6 AM)
* • Extreme PCA Values: Outliers in principal components indicate anomalies
* • Rapid Sequences: Multiple transactions in short time windows

## 6.2 Risk Scoring Framework

Composite risk scoring combines multiple fraud indicators:

# Domain-specific risk indicators  
df['Amount\_very\_high'] = (df['Amount'] > df['Amount'].quantile(0.99)).astype(int)  
df['Amount\_very\_low'] = (df['Amount'] < 1).astype(int)  
df['Time\_unusual\_hours'] = ((df['Hour'] >= 3) & (df['Hour'] <= 6)).astype(int)  
  
# Composite risk score  
risk\_features = ['Amount\_very\_high', 'Amount\_very\_low', 'Time\_unusual\_hours']  
df['Risk\_score'] = df[risk\_features].sum(axis=1)  
df['Risk\_score\_normalized'] = df['Risk\_score'] / len(risk\_features)

## 6.3 PCA Interaction Features

Advanced interactions between PCA components reveal hidden patterns:

* • Multiplicative interactions between top PCA features
* • Additive combinations revealing cumulative effects
* • Ratio calculations for relative importance
* • Statistical aggregations (sum, mean, std, skew, kurtosis)
* • Count-based features (positive, negative, zero values)
* • Extreme value indicators for anomaly detection

# 7. Data Visualization Strategy

## 7.1 Comprehensive Visualization Suite

Advanced visualizations provide deep insights into data patterns:

* • Class Distribution Plots: Bar charts and pie charts showing fraud vs. normal ratios
* • Correlation Heatmaps: Matrix visualizations of feature relationships
* • Distribution Comparisons: Overlaid histograms comparing fraud vs. normal
* • Temporal Analysis Plots: Time-series and hourly pattern visualizations
* • Amount Analysis Charts: Transaction value distributions and patterns
* • Feature Importance Plots: Ranking visualizations for predictive features

## 7.2 Interactive Analysis Capabilities

Visualization framework supports interactive exploration:

* • Automated plot generation and saving
* • Configurable visualization parameters
* • Statistical overlay on distribution plots
* • Correlation threshold filtering
* • Time-based aggregation options
* • Feature selection for focused analysis

# 8. Feature Selection & Optimization

## 8.1 Automated Feature Selection Pipeline

Sophisticated feature selection ensures optimal model performance:

* • Mutual Information: Information-theoretic approach to feature relevance
* • Statistical Tests: F-statistics and chi-square tests for significance
* • Correlation Filtering: Removing highly correlated redundant features
* • Constant Feature Removal: Eliminating features with no variance
* • Domain Validation: Expert review of selected features

## 8.2 Feature Selection Implementation

def select\_best\_features(self, X, y, k=50):  
 """Select best features using statistical tests."""  
   
 # Remove constant features  
 constant\_features = X.columns[X.nunique() <= 1].tolist()  
 X = X.drop(columns=constant\_features)  
   
 # Feature selection using mutual information  
 selector = SelectKBest(score\_func=mutual\_info\_classif, k=min(k, X.shape[1]))  
 X\_selected = selector.fit\_transform(X, y)  
   
 # Get selected feature names  
 selected\_features = X.columns[selector.get\_support()].tolist()  
   
 return pd.DataFrame(X\_selected, columns=selected\_features), selected\_features

# 9. Technical Implementation Details

## 9.1 Modular Architecture Design

The EDA and feature engineering modules follow clean architecture principles:

* • AdvancedEDA Class: Comprehensive analysis with statistical testing
* • AdvancedFeatureEngineer Class: Systematic feature creation pipeline
* • Visualization Engine: Automated plot generation and customization
* • Feature Selection Pipeline: Automated selection and optimization
* • Quality Validation: Data integrity and feature validation checks

## 9.2 Performance Optimization

Implementation includes several performance optimizations:

* • Vectorized operations using NumPy and Pandas
* • Memory-efficient feature creation with chunking
* • Parallel processing for statistical computations
* • Lazy evaluation for large dataset handling
* • Caching mechanisms for repeated calculations
* • Progressive feature selection to manage complexity

## 9.3 Error Handling and Robustness

Comprehensive error handling ensures reliable operation:

* • Graceful handling of missing or invalid data
* • Division by zero protection in ratio calculations
* • Memory overflow prevention for large datasets
* • Feature validation and type checking
* • Fallback mechanisms for failed computations
* • Comprehensive logging for debugging

# 10. Quality Assurance & Validation

## 10.1 Data Quality Validation

Comprehensive validation ensures data quality throughout the pipeline:

* • Missing value detection and handling strategies
* • Outlier identification using statistical methods
* • Data type consistency validation
* • Range validation for numerical features
* • Duplicate detection and removal
* • Class distribution monitoring

## 10.2 Feature Validation Framework

Created features undergo rigorous validation:

* • Statistical significance testing for new features
* • Correlation analysis to prevent redundancy
* • Distribution analysis for feature stability
* • Business logic validation for domain features
* • Performance impact assessment
* • Interpretability evaluation

# 11. Results & Key Insights

## 11.1 Statistical Discoveries

Advanced EDA revealed critical insights about fraud patterns:

* • Significant differences in transaction amounts between fraud and normal
* • Temporal patterns showing increased fraud during specific hours
* • PCA feature correlations indicating anomaly detection potential
* • Class imbalance requiring sophisticated resampling strategies
* • Feature interactions revealing hidden fraud indicators

## 11.2 Feature Engineering Outcomes

Feature engineering process generated comprehensive feature sets:

* • 100+ sophisticated features created from original 30 features
* • Temporal features capturing time-based fraud patterns
* • Amount transformations handling skewed distributions
* • PCA interactions revealing complex relationships
* • Domain-specific risk indicators based on fraud expertise
* • Automated feature selection identifying top predictive features

## 11.3 Visualization Insights

Comprehensive visualizations provided actionable insights:

* • Clear separation between fraud and normal transaction patterns
* • Correlation heatmaps identifying feature relationships
* • Temporal analysis revealing peak fraud periods
* • Amount distribution analysis showing fraud characteristics
* • Feature importance rankings guiding model development

# 12. Next Steps & Integration

## 12.1 Model Training Integration

Stage 2 outputs seamlessly integrate with machine learning pipeline:

* • Enhanced feature sets ready for model training
* • Statistical insights informing model selection
* • Feature importance rankings guiding hyperparameter tuning
* • Data quality validation ensuring robust training
* • Visualization tools for model interpretation

## 12.2 Stage 3 Preparation

Foundation established for advanced model training:

* • Comprehensive feature sets for ensemble model training
* • Statistical baselines for model performance comparison
* • Feature selection pipelines for optimization
* • Visualization frameworks for model evaluation
* • Domain insights for model interpretation

## 12.3 Continuous Improvement Framework

Established framework supports ongoing enhancement:

* • Modular design allows easy feature addition
* • Automated validation ensures quality maintenance
* • Performance monitoring identifies optimization opportunities
* • Documentation supports knowledge transfer
* • Version control enables reproducible analysis

# Conclusion

Stage 2 has successfully established a comprehensive data science foundation for the fraud detection system. Through advanced exploratory data analysis and sophisticated feature engineering, we have transformed raw transaction data into a rich, informative feature space that captures the complex patterns indicative of fraudulent behavior.

The systematic approach to statistical analysis, domain-driven feature creation, and automated optimization ensures that our machine learning models will have access to the highest quality predictive features. The comprehensive visualization suite and validation frameworks provide the tools necessary for ongoing model development and interpretation.

This foundation positions the project for successful advanced model training in Stage 3, with the confidence that our feature engineering has captured the essential patterns needed for high-performance fraud detection.