Multilingual ISIC Code Classification System for Labour Force Survey Data Collection

An Automated Machine Learning Solution for Statistical Operations in Rwanda

---

Report Prepared By:
NGIRINSHUTI Fidele, Lead Data Scientist, model development, feature engineering, algorithm selection, and performance optimization

UMUHOZA Marie Ange, Data preprocessing, exploratory analysis, model evaluation, and validation

HATEGEKIMANA Emelyne, Streamlit application development, UI/UX design, and deployment

Institution:
National Institute of Statistics of Rwanda (NISR)

Date: January 2026

Executive Summary

This report presents an automated classification system developed to assign International Standard Industrial Classification (ISIC) codes to business descriptions collected during Rwanda's Labour Force Survey operations. The system employs machine learning techniques optimized for multilingual text processing, supporting Kinyarwanda, English, and French inputs.

The developed solution addresses a critical operational bottleneck: the manual classification of over 126,000 business descriptions, a process that traditionally requires months of labour-intensive work by trained statistical coders. Through advanced natural language processing and feature engineering, the system achieves 76.51% classification accuracy across 396 distinct ISIC codes.

Key Achievements:

- **Accuracy**: 76.51% on held-out test data (38,046 samples)
- **Efficiency**: Processes classifications in under 100 milliseconds per sample
- **Coverage**: 68% of predictions achieve high confidence (>70%), suitable for automated coding
- **Impact**: Potential to reduce manual coding workload by 68%, translating to significant time and cost savings

The system has been deployed as an interactive web application, enabling both single-text predictions and batch processing of survey datasets, with built-in confidence scoring to support quality assurance workflows.

1. Introduction and Problem Statement

1.1 Background Context

The National Institute of Statistics of Rwanda (NISR) conducts periodic Labour Force Surveys (LFS) to collect comprehensive data on employment, unemployment, and economic activity across the country. These surveys are fundamental instruments for evidence-based policy making, supporting decisions in areas including:

- Employment and labor market policy development
- Economic planning and industrial strategy
- Human resource development and skills training
- Social protection program design
- International reporting obligations (ILO, UN, African Union)

A critical component of LFS data processing is the classification of respondents' economic activities into standardized International Standard Industrial Classification (ISIC) codes. These codes provide a coherent framework for categorizing productive economic activities, enabling:

- **International comparability** of economic statistics
- **Longitudinal analysis** of sectoral employment trends
- **Policy targeting** based on specific industries or economic activities
- **Resource allocation** informed by sectoral economic performance

1.2 The Problem

Traditional ISIC coding in Rwanda's LFS operations follows a manual process:

1. Survey enumerators collect free-text descriptions of business activities from respondents in their preferred language (Kinyarwanda, English, or French)
2. These descriptions are compiled into datasets containing tens of thousands of responses
3. Trained statistical coders manually read each description and assign the appropriate 4-digit ISIC code
4. Supervisors review a sample of codes for quality assurance
5. Disputed or ambiguous cases are escalated for expert review

This process faces several critical challenges:

Consistency and Quality
Human coders, despite training, exhibit variability in interpretation. Studies in similar statistical offices have documented inter-coder agreement rates of 75-85% for occupation and industry coding, meaning that 15-25% of identical descriptions receive different codes depending on which coder processes them. This inconsistency undermines data quality and affects trend analysis.

Multilingual Complexity
Rwanda's trilingual survey environment presents unique challenges. Coders must be proficient in Kinyarwanda, English, and French, and must recognize that the same economic activity may be described using different terminology across languages. For example:

- "Ubuhinzi bw'ibigori" (Kinyarwanda)
- "Cereal farming" (English)
- "Culture de céréales" (French)

All refer to the same ISIC code but require linguistic and domain expertise to classify correctly.

Training Requirements
New coders require extensive training on:

- The hierarchical structure of ISIC Rev. 4 (21 sections, 88 divisions, 238 groups, 419 classes)
- Industry-specific terminology in three languages
- Coding conventions for ambiguous cases
- Quality control procedures

1.3 Institutional Importance

For NISR:
Automating ISIC coding directly addresses strategic priorities outlined in the National Strategy for the Development of Statistics (NSDS) 2024-2029, particularly:

- Goal 2: Improve efficiency and timeliness of statistical production
- Goal 3: Modernize data collection and processing systems
- Goal 4: Build institutional capacity for data management

For International Credibility:
Maintaining high-quality, timely labour statistics is essential for Rwanda's reporting obligations to international organizations (ILO, World Bank, UN) and affects country ratings, investment climate assessments, and development partner relationships.

1.4 Project Objectives

This project was initiated to develop an automated classification system with the following objectives:

Primary Technical Objectives:

1. Build a machine learning classifier capable of predicting 4-digit ISIC codes from free-text business descriptions
2. Optimize the system for multilingual input (Kinyarwanda, English, French) without requiring language identification
3. Achieve classification accuracy sufficient to reduce manual coding workload while maintaining data quality standards
4. Provide confidence scores for each prediction to enable risk-based quality assurance

Operational Objectives:

1. Reduce manual coding time by at least 50%
2. Improve consistency by eliminating inter-coder variability for automatically coded cases
3. Enable faster survey data release, reducing processing time from 6 months to 4 months
4. Create an accessible interface requiring minimal technical training for statistical office staff

Institutional Objectives:

1. Build NISR's capacity in machine learning applications for statistical production
2. Establish a framework for continuous model improvement through feedback loops
3. Document methodology and code to enable knowledge transfer and replication
4. Create a foundation for applying similar approaches to occupation coding, expenditure classification, and other survey coding tasks

1.5 Scope of the Project

In Scope:

- Classification of business activity descriptions into ISIC Rev. 4 codes at the 4-digit class level
- Support for Kinyarwanda, English, and French text inputs (including mixed-language descriptions)
- Single-text prediction interface for ad-hoc coding needs
- Batch CSV processing for full survey dataset coding

- Confidence scoring for quality assurance and human review prioritization
- Web-based application deployable on NISR infrastructure

Out of Scope:

- Real-time classification during field data collection (to be addressed in future phases)
- Integration with existing survey management systems (requires separate IT project)
- Multi-label classification for businesses with multiple distinct activities
- Classification beyond ISIC Rev. 4 structure (e.g., occupation codes, product codes)
- Language translation or description enhancement capabilities

---

2. Data Description and Processing

2.1 Data Sources

The training dataset consists of business activity descriptions and corresponding ISIC codes from Rwanda's 2021 Labour Force Survey, the most recent comprehensive employment survey conducted by NISR.

Dataset Characteristics:

Attribute	Value
Total Records	126,817 business descriptions
Source Survey	Rwanda Labour Force Survey 2021
Collection Period	August - October 2021
Geographic Coverage	All 30 districts of Rwanda
Sampling Frame	Household-based sampling
Data Format	CSV file with ISO-8859-1 encoding

Key Variables:

D03B (Business Description Text):

- Free-text field capturing respondent's description of their main economic activity
- Variable length: 5 to 500 characters (mean: 42 characters)
- Language: Mixed Kinyarwanda, English, and French (no language identifier in data)
- Collection method: Enumerator transcription of verbal responses
- Examples:
    - "Ubuhinzi bw'ibigori n'ubworozi bw'amatungo" (Cereal farming and livestock)
    - "Retail shop selling food items" (Retail trade)
    - "Construction de bâtiments" (Building construction)

D03B1 (ISIC Code):

- 4-digit ISIC Rev. 4 classification code
- Range: 0111 to 9900 (though stored as integers: 111 to 9900)
- Originally assigned by trained NISR coders following standard coding procedures
- Inter-coder reliability tested on 10% subsample: 78% exact agreement
- Quality assurance: Supervisory review of 5% of coded cases

Additional Context Variables (not used in current model):

- District code (administrative location)
- Urban/rural indicator
- Business size (number of employees)
- Respondent education level

2.2 Data Quality Assessment

Initial Data Quality Issues Identified:

1. **Missing Values**
    - 2,847 records (2.2%) had empty business description field
    - 1,124 records (0.9%) had missing ISIC codes
    - Total: 3,971 records (3.1%) with missing data
2. **Encoding Issues**
    - 423 records contained character encoding errors (e.g., "Ã©" instead of "é")
    - Primarily affected French-language responses
    - Required Unicode normalization
3. **Class Imbalance**
    - Top 10 ISIC codes represented 62% of all records
    - 127 ISIC codes (32%) had only 1 training sample
    - 43 ISIC codes (11%) had 2-5 samples
4. **Inconsistent ISIC Code Assignment**
    - Manual review of 500 random cases revealed:
        - 8% potentially miscoded (coder error or ambiguous description)
        - 12% could legitimately be assigned to multiple codes (borderline cases)
        - 3% assigned to deprecated ISIC codes from earlier revisions
5. **Description Quality Variability**
    - Highly informative: "Growing maize and beans for commercial sale" (32% of cases)
    - Moderately informative: "Small business selling various items" (45% of cases)
    - Vague: "Business", "Trading", "Work" (23% of cases)

2.3 Data Cleaning and Pre-processing

The following cleaning steps were applied to prepare the dataset for model training:

Step 1: Handling Missing Values

```
# Remove records with missing business descriptions or ISIC codes
data = data.dropna(subset=['D03B', 'D03B1'])
# Result: Dataset reduced from 126,817 to 122,846 records (3.1% removed)
```

Step 2: Unicode Normalization All text was normalized to handle character encoding inconsistencies:

```
# Convert to Unicode NFC form and strip leading/trailing whitespace
data['D03B'] = data['D03B'].str.normalize('NFC').str.strip()
# Removes encoding artifacts and standardizes accented characters
```

Rationale: Ensures that "développement" and "dÃ©veloppement" are treated identically.

Step 3: ISIC Code Standardization

```
# Ensure ISIC codes are 4-digit integers
data['D03B1'] = data['D03B1'].astype(int)
# Validate against official ISIC Rev. 4 code list
valid_codes = load_isic_rev4_codes()
data = data[data['D03B1'].isin(valid_codes)]
# Result: 1,237 records removed with invalid/deprecated codes
```

Step 4: Minimum Sample Size Filtering To enable stratified train-test splitting, ISIC codes with only 1 sample were removed:

```
# Count samples per ISIC code
code_counts = data['D03B1'].value_counts()
# Keep only codes with 2+ samples
valid_codes = code_counts[code_counts >= 2].index
data = data[data['D03B1'].isin(valid_codes)]
# Result: 127 singleton codes removed, final dataset: 121,609 records
```

Rationale: Stratified splitting requires at least 2 samples per class to ensure both train and test sets contain examples of each code.

Step 5: Text Cleaning (Minimal) Unlike many NLP tasks, minimal text cleaning was applied to preserve linguistic features:

```
# NO lowercasing (preserve proper nouns and language-specific capitalization)
# NO stopword removal (may contain meaningful ISIC distinctions)
# NO stemming/lemmatization (important for Kinyarwanda morphology)
# Only: Remove excessive whitespace
data['D03B'] = data['D03B'].str.replace(r'\s+', ' ', regex=True)
```

Rationale: Aggressive text cleaning can remove valuable information for classification. For example:

- "retail TRADE" vs. "retail trade association" (different ISIC codes)
- Kinyarwanda prefixes indicate semantic categories (e.g., "ubuhinzi" = farming as abstract noun)

2.4 Final Dataset Summary

After cleaning and preprocessing:

Metric	Value
Total Training Samples	121,609
Unique ISIC Codes	396
Average Description Length	42 characters (8.3 words)
Minimum Description Length	5 characters
Maximum Description Length	487 characters
Language Distribution	Mixed (no labels available)

ISIC Code Distribution:

Category	Count	Percentage
High-frequency codes (>1,000 samples)	8 codes	2.0%
Medium-frequency codes (100-1,000 samples)	78 codes	19.7%
Low-frequency codes (10-99 samples)	187 codes	47.2%
Very low-frequency codes (2-9 samples)	123 codes	31.1%

Top 10 Most Frequent ISIC Codes:

ISIC Code	Description	Count	Percentage
9700	Activities of households as employers	16,234	13.3%
111	Growing of cereals and other crops	12,418	10.2%
4100	Construction of buildings	8,932	7.3%
4781	Retail sale via stalls and markets	7,215	5.9%
113	Growing of vegetables and fruits	6,891	5.7%
4921	Urban and suburban passenger transport	5,324	4.4%
4711	Retail sale in non-specialized stores	4,782	3.9%
8510	Pre-primary and primary education	3,654	3.0%
5630	Beverage serving activities	2,987	2.5%
5610	Restaurants and mobile food service	2,341	1.9%

2.5 Train-Test Split Strategy

Stratified Random Sampling:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.30,      # 70% train, 30% test
    random_state=42,     # Reproducibility
    stratify=y           # Maintain class distribution
)
```

Resulting Split:

Set	Samples	ISIC Codes	Percentage
Training	85,126	396	70%
Testing	36,483	396	30%

Rationale for 70/30 Split:

- **70% training**: Provides sufficient data for model learning, especially for rare ISIC codes
- **30% testing**: Large enough to provide reliable performance estimates across all 396 codes
- **Stratification**: Ensures rare codes appear in both sets, preventing zero-sample classes in test set

3. Methodology

3.1 Overall Approach

The classification system employs a supervised machine learning pipeline consisting of:

1. **Feature Extraction**: Converting text into numerical representations
2. **Model Training**: Learning patterns from labeled examples
3. **Prediction**: Assigning ISIC codes to new business descriptions
4. **Confidence Scoring**: Estimating prediction reliability

The approach prioritizes:

- **Interpretability**: Statistical office staff must understand how predictions are made
- **Efficiency**: Fast inference suitable for batch processing large datasets

- **Maintainability**: Simple architecture enabling future updates by non-ML experts
- **Multilingual Robustness**: Features that work across Kinyarwanda, English, and French

3.2 Feature Engineering

Traditional word-based text representations face challenges with multilingual, morphologically rich text. The solution employs a **dual-feature extraction strategy** combining word-level and character-level patterns.

3.2.1 Word-Level TF-IDF Features

Term Frequency-Inverse Document Frequency (TF-IDF) measures word importance by balancing:

- **Term Frequency (TF)**: How often a word appears in a document
- **Inverse Document Frequency (IDF)**: How rare the word is across all documents

Implementation:

```
from sklearn.feature_extraction.text import TfidfVectorizer

word_tfidf = TfidfVectorizer(
    max_features=3000,        # Keep top 3,000 most important words
    min_df=2,                 # Word must appear in at least 2 documents
    ngram_range=(1, 2),       # Unigrams and bigrams
    analyzer='word',          # Token at word boundaries
    strip_accents='unicode',  # Normalize accented characters
    lowercase=True            # Case-insensitive matching
)
```

Captured Patterns:

- Single words: "agriculture", "construction", "retail"
- Bigrams (2-word phrases): "retail trade", "food production", "passenger transport"

Example:

- Input: "Growing maize and beans for commercial sale"
- Top TF-IDF features: "maize" (0.84), "beans" (0.82), "growing maize" (0.76), "commercial" (0.61)

Why This Works:

- Domain-specific terminology (e.g., "cereal", "construction", "retail") strongly indicates ISIC category
- Bigrams capture multi-word industry terms
- IDF weighting reduces influence of common words like "business", "work", "doing"

3.2.2 Character-Level TF-IDF Features

Character n-grams capture subword patterns crucial for Kinyarwanda, which uses agglutinative morphology where meaning is built through prefixes and suffixes.

Implementation:

```
char_tfidf = TfidfVectorizer(
    max_features=2000,        # Keep top 2,000 character patterns
```

```
    analyzer='char',           # Token at character level
    ngram_range=(2, 5),        # Character sequences of length 2-5
    strip_accents='unicode',   # Normalize accented characters
    lowercase=True             # Case-insensitive
)
```

Captured Patterns:

- 2-grams: "ub", "hi", "nz" (from "ubuhinzi" = farming)
- 3-grams: "ubu", "buh", "uhi", "hin", "inz", "nzi"
- 4-grams: "ubuh", "buhi", "uhin", "hinz", "inzi"
- 5-grams: "ubuhi", "buhin", "uhinz", "hinzi"

Example (Kinyarwanda):

- Input: "Ubuhinzi bw'ibigori" (Cereal farming)
- Top character n-grams: "ubuhi" (0.92), "buhin" (0.89), "hinzi" (0.87), "igori" (0.81)

These patterns help the model recognize that words containing "ubuhinzi" or "uburozi" (morphological variants of farming/livestock) belong to agricultural ISIC codes, even if the exact word form wasn't in training data.

Why This Works for Multilingual Text:

- **Kinyarwanda morphology**: Captures prefix patterns indicating semantic categories
- **French/English cognates**: Recognizes shared roots (e.g., "constru" appears in "construction" and "construcción")
- **Spelling variations**: Robust to minor typos or transliteration differences
- **Language-agnostic**: Doesn't require knowing which language is being processed

3.2.3 Feature Combination

The two feature sets are combined using scikit-learn's `FeatureUnion`:

```
from sklearn.pipeline import FeatureUnion

combined_features = FeatureUnion([
    ('word_features', word_tfidf),    # 3,000 word-level features
    ('char_features', char_tfidf)     # 2,000 character-level features
])
# Result: ~5,000 combined features per document
```

Feature Space Properties:

- **Dimensionality**: ~5,000 features (sparse representation)
- **Sparsity**: Each document has typically 10-30 non-zero features
- **Memory efficiency**: Sparse matrix storage (CSR format)
- **Complementarity**: Word features capture semantics, character features capture morphology

Rationale: This dual approach leverages strengths of both representations:

- Word-level features work well for English/French with consistent word boundaries
- Character-level features excel for Kinyarwanda's morphological complexity

- Together, they provide robust multilingual coverage

3.3 Model Selection and Training

Three classification algorithms were evaluated to identify the best performer:

3.3.1 Logistic Regression

Algorithm Overview: Logistic regression estimates the probability that a text belongs to each ISIC code by learning a weighted combination of features. Despite its name, it's a classification algorithm that models:

$P(\text{ISIC code} \mid \text{text features}) = \text{softmax}(w_1x_1 + w_2x_2 + ... + w_nx_n + b)$

Configuration:

```
from sklearn.linear_model import LogisticRegression

LogisticRegression(
    max_iter=2000,           # Allow 2000 iterations for convergence
    C=1.0,                   # Regularization strength (higher = less
regularization)
    class_weight='balanced', # Weight classes inversely proportional to
frequency
    random_state=42,         # Reproducibility
    solver='lbfgs'          # Limited-memory BFGS optimizer
)
```

Advantages:

- Fast training and inference
- Interpretable: Feature weights show which words/patterns predict each code
- Good baseline for text classification
- Probabilistic outputs for confidence scoring

Disadvantages:

- Assumes linear separability
- May underfit complex patterns
- Performance limited by feature engineering quality

3.3.2 Random Forest

Algorithm Overview: Random Forest builds an ensemble of decision trees, each trained on a random subset of features and samples. Final prediction is made by majority vote.

Configuration:

```
from sklearn.ensemble import RandomForestClassifier

RandomForestClassifier(
    n_estimators=200,        # Build 200 decision trees
    max_depth=25,            # Limit tree depth to prevent overfitting
```

```
    min_samples_split=5,     # Require 5+ samples to split a node
    class_weight='balanced', # Handle class imbalance
    random_state=42,         # Reproducibility
    n_jobs=-1               # Use all CPU cores for parallel training
)
```

Advantages:

- Captures non-linear patterns
- Robust to noisy features
- Feature importance rankings for interpretability
- Handles high-dimensional data well

Disadvantages:

- Slower inference than linear models
- Larger model size (memory requirements)
- Can overfit with small training samples

3.3.3 K-Nearest Neighbors (KNN)

Algorithm Overview: KNN classifies new texts by finding the K most similar training examples (using cosine similarity) and taking a weighted vote of their labels. Closer neighbors have more influence.

Configuration:

```
from sklearn.neighbors import KNeighborsClassifier

KNeighborsClassifier(
    n_neighbors=7,          # Use 7 nearest neighbors
    weights='distance',     # Weight by inverse distance
    metric='cosine'         # Cosine similarity for text
)
```

Advantages:

- No explicit training phase (instance-based learning)
- Naturally handles multimodal distributions
- Intuitive: "Similar descriptions → Same ISIC code"
- Excellent for high-dimensional sparse data (like TF-IDF)

Disadvantages:

- Slower inference (must compare to all training samples)
- Memory-intensive (stores entire training set)
- Sensitive to feature scaling

Why Cosine Similarity? For text represented as TF-IDF vectors, cosine similarity is ideal:

- Measures angle between vectors, not Euclidean distance
- Insensitive to document length
- Range [0, 1] where 1 = identical, 0 = completely different

3.4 Handling Class Imbalance

All models were configured with `class_weight='balanced'` to address the severe imbalance in ISIC code frequencies.

Standard Loss Function: Misclassifying any sample incurs equal penalty.

Balanced Loss Function: Penalty for misclassifying rare codes is increased proportionally:

weight(ISIC code) = total_samples / (num_classes × samples_in_class)

Example:

- ISIC 9700 (16,234 samples): weight = 121,609 / (396 × 16,234) = 0.019
- ISIC 3314 (2 samples): weight = 121,609 / (396 × 2) = 153.3

Misclassifying the rare code incurs 8,000× higher penalty, forcing the model to learn patterns for rare codes despite limited examples.

3.5 Model Training Pipeline

Complete training pipeline implemented using scikit-learn:

```
from sklearn.pipeline import Pipeline

# Example: KNN Pipeline
knn_pipeline = Pipeline([
    ('features', combined_features),  # Feature extraction
    ('classifier', KNeighborsClassifier(...))  # Classification
])

# Training
knn_pipeline.fit(X_train['D03B'], y_train)

# Prediction
predictions = knn_pipeline.predict(X_test['D03B'])
probabilities = knn_pipeline.predict_proba(X_test['D03B'])
```

Training Computational Requirements:

Model	Training Time	Memory Usage	Model Size
Logistic Regression	3.2 minutes	1.8 GB	42 MB
Random Forest	28.7 minutes	4.5 GB	387 MB
KNN	0.8 minutes	2.1 GB	623 MB

Hardware: Intel i7-11800H, 16GB RAM

3.6 Evaluation Metrics

Primary Metric: Accuracy

```
Accuracy = (Correct Predictions) / (Total Predictions)
```

Per-Class Metrics:

- **Precision**: Of all predictions for ISIC code X, what percentage were correct?
- **Recall**: Of all actual ISIC code X samples, what percentage were correctly identified?
- **F1-Score**: Harmonic mean of precision and recall

Model Health Metrics:

- **Training Accuracy**: Performance on training set (checks for underfitting)
- **Test Accuracy**: Performance on held-out test set (true generalization)
- **Overfitting Gap**: Training accuracy - Test accuracy (checks for overfitting)

Confidence Calibration: For each prediction, the model's maximum predicted probability is used as confidence score:

$$\text{Confidence} = \max(P(\text{ISIC}_1), P(\text{ISIC}_2), \ldots, P(\text{ISIC}_{396}))$$

3.7 Tools and Technologies

Programming Language:

- Python 3.13 (chosen for rich ecosystem of data science libraries)

Core Libraries:

- **scikit-learn 1.7.2**: Machine learning algorithms and pipelines
- **pandas 2.2.3**: Data manipulation and analysis
- **NumPy 1.24.3**: Numerical computing
- **joblib 1.5.2**: Model serialization and parallel processing

Web Application:

- **Streamlit 1.28.0**: Interactive web interface framework
- **Custom CSS**: Enhanced user experience

Development Environment:

- **Jupyter Notebook**: Interactive model development and experimentation
- **Git**: Version control
- **VS Code**: Code editor

Deployment Infrastructure:

- **Local Server**: Initial deployment on NISR infrastructure
- **Future**: Streamlit Community Cloud or containerized deployment (Docker)

3.8 Rationale for Design Decisions

Why TF-IDF Instead of Deep Learning?

- **Interpretability**: Statistical office requires explainable predictions
- **Efficiency**: Fast inference suitable for batch processing
- **Data Requirements**: Deep learning requires 10-100× more training data
- **Maintainability**: NISR staff can update TF-IDF models without ML PhDs
- **Resource Constraints**: No GPU infrastructure required

Why Multiple Models? Testing multiple algorithms ensures the best performer is selected based on empirical evidence rather than assumptions. Comparative evaluation builds confidence in the chosen solution.

Why 70/30 Split?

- Larger test set (30%) provides more reliable performance estimates
- With 121,609 samples, even 70% training (85,126 samples) is sufficient
- Stratification ensures all 396 ISIC codes appear in both sets

Why Cosine Similarity for KNN? Cosine similarity is standard for sparse, high-dimensional text features because it measures semantic similarity regardless of document length.

---

4. Results and Outputs

4.1 Model Performance Comparison

All three models were trained and evaluated on the same train-test split. The following table presents comparative performance:

Model	Training Accuracy	Test Accuracy	Overfitting Gap	Training Time
K-Nearest Neighbors	**89.13%**	**76.51%**	**12.62%**	0.8 min
Logistic Regression	72.46%	68.30%	4.16%	3.2 min
Random Forest	74.59%	65.12%	9.48%	28.7 min

Winner: K-Nearest Neighbors (KNN)

- **Highest test accuracy**: 76.51% correctly classified business descriptions
- **Strong training accuracy**: 89.13% indicates the model learned patterns well
- **Acceptable overfitting**: 12.62% gap is reasonable given model complexity
- **Fast training**: Completes in under 1 minute

Key Findings:

1. **KNN outperforms other models** despite being conceptually simpler
2. **Logistic Regression shows least overfitting** (4.16% gap) but lower overall accuracy
3. **Random Forest underperforms** despite higher training accuracy—likely overfitting despite regularization

4.2 Detailed Performance Analysis: KNN Model

4.2.1 Overall Metrics

Test Set Performance (38,046 samples):

Overall Accuracy: 76.51%
Macro Average Precision: 0.34
Macro Average Recall: 0.31
Macro Average F1-Score: 0.31

Weighted Average Precision