جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY

Computation Project

Team Members

Name	ID
حمزة حسين يوسف عمران	22011501
احمد خليفة عبدالرؤوف محمد	22010018
كريم محمد سامي ابوشادي	22010378
معاذ مصطفى عبدالحميد مصطفى	22010263
عبد الرحمن هشام رجب ابراهيم	22010136

1. Introduction:

In real-world classification tasks, especially in domains like finance and security, distinguishing between authentic and forged data is crucial. The Banknote Authentication dataset, provided by the UCI Machine Learning Repository, serves as a practical benchmark for evaluating machine learning models in such contexts. The dataset comprises 1,372 grayscale images of banknotes—both genuine and forged—represented by four numerical features derived from wavelet-transformed images: variance, skewness, curtosis, and entropy.

This project focuses on training a Support Vector Machine (SVM) from scratch using a non-differentiable loss function—Hinge Loss. Unlike standard gradient-based optimization methods, Hinge Loss necessitates techniques such as sub-gradient descent, which are better suited for non-smooth optimization landscapes. By comparing Gradient Descent and Sub-gradient Descent, we aim to study the trade-offs in terms of accuracy, convergence, and generalization ability.

---

2. Objectives:

· **Dataset Preparation**

- Load and preprocess the **Banknote Authentication** dataset from the UCI Repository.
- Perform train-validation-test split and apply scaling techniques to standardize features.

- **Model Implementation**
    - Build an **SVM classifier from scratch** utilizing **Hinge Loss**.
    - Implement two training strategies: **Gradient Descent** and **Sub-gradient Descent**.
- **Model Training and Evaluation**
    - Train both optimization methods over multiple epochs.
    - Track and record **loss** and **accuracy** at each epoch on training and validation sets.
- **Visualization and Analysis**
    - Plot **loss curves** and **accuracy curves** for both training strategies.
    - Generate **confusion matrices** to evaluate classification performance.
    - Compare both methods in terms of:
        - Classification accuracy
        - Convergence speed
        - Stability of the loss function
        - Generalization performance on test data

---

1. Exploratory Data Analysis (EDA)

1. Dataset Overview

1.1 Importing Libraries

Essential libraries for data analysis, visualization, preprocessing, and modeling were imported. This includes pandas, numpy, matplotlib, seaborn, and sklearn.

1.2 Dataset Introduction

Dataset: churn_prediction.csv
Contains customer banking data with features on demographics, transactions, and balances.

1.3 Key Attributes

- customer_id: Unique customer identifier
- churn: Target variable (1 = churned, 0 = not churned)

- Financial metrics: current_balance, average_monthly_balance_prevQ, credit, debit, etc.
- Demographics: age, gender, occupation, city, dependents
- Temporal: vintage, last_transaction

2. Exploratory Data Analysis (EDA)

2.1 Data Summary

- Shape: (27620, 21)
- Churn Rate: 18.7%
- Demographics: Avg. age ≈ 48, most have 0 dependents, gender mostly Male
- City/Branch: High cardinality
- Balances: Large variance; requires scaling/log-transform
- Outliers: Extremely high balances and dependents
- Missing Data: Found in gender, dependents, occupation, city

2.2 Data Cleaning

- Filled missing: gender → "Unknown", dependents → median, occupation → mode, city → -1
- Converted: last_transaction → datetime, churn → categorical
- Removed customers with age < 18
- Dropped duplicates

3. Data Visualization and Insights

3.1 Correlation Analysis

- Weak correlation between churn and numeric variables
- High redundancy: current_balance ↔ previous_month_end_balance (0.95), average_monthly_balance_prevQ ↔ current_balance (0.96)

3.2 Categorical Features

- Gender: Males dominate; slightly higher churn
- Occupation: Most are Self-employed → Highest churn; Salaried next highest

3.3 Churn Evolution Over Time

- Sharp increase in churn after October 2019

3.4 Age Distribution

- Peak churn: Age 30–45

3.5 City-wise Churn

- City 1020: Highest in both customer count and churn

4. Summary of Key Findings

Aspect	Insights
Churn Rate	18.7% – significant, requiring strategic retention
High Churn Segments	Self-employed, males, ages 30–45
Outliers & Skewness	Large transaction values; requires transformation
Redundant Features	High correlation among balance-related fields
Temporal Pattern	Churn rose rapidly in late 2019
City Impact	A few cities contribute disproportionately to churn

---

2.Feature Selection

1. Chi-Square Feature Selection

Categorical features were evaluated using the Chi-Square test to measure their association with the target variable 'churn'. Features were encoded and tested using scikit-learn's SelectKBest.
Top features by Chi-Square score:
- branch_code: 26694.20
- city: 103.88
- dependents: 59.63
- occupation, gender, customer_nw_category: low scores
Interpretation: 'branch_code' has the strongest association with churn.

2. ANOVA Feature Selection

Numerical features were assessed using the ANOVA F-test to identify significant relationships with churn. Missing values were filled with 0, and scikit-learn's f_classif was used.
Top features by ANOVA score:
- previous_month_debit: 143.14
- current_month_debit: 60.80
- previous_month_credit: 48.40
Interpretation: Transaction activity is highly associated with churn.

3. Information Gain Feature Selection

Mutual Information (Information Gain) was computed to evaluate feature importance, capturing both linear and non-linear relationships.
Top features by Information Gain:
- current_balance: 0.0815
- current_month_balance: 0.0409
- current_month_debit: 0.0324
Interpretation: Balance and debit-related features are most informative.

4. Pearson Correlation Analysis

Linear correlation between numerical features and churn was measured using the Pearson correlation coefficient.
Top correlated features (positive):
- previous_month_debit: 0.0719
- current_month_debit: 0.0469
Top negatively correlated features:
- age: -0.0314
- current_balance: -0.0266
Interpretation: All correlations are weak, but transaction features show slight associations.

5. Forward Selection using AIC

Logistic regression and AIC were used to iteratively select features that improved model fit. Features like 'current_balance', 'branch_code', and 'age' were selected early.
Final selected features (sample):
- current_balance, branch_code, age, current_month_debit, etc.
Interpretation: AIC-guided selection led to a strong subset of 12 features despite convergence warnings.

6. Backward Elimination using AIC

Started with all features and removed those that did not improve AIC. Final model retained 16 features including age, branch_code, and transaction-related variables.
Interpretation: The resulting model is more concise and interpretable, with reduced noise.

Feature Extraction with PCA and LDA

Objective

The goal of this task was to apply **Principal Component Analysis (PCA)** and **Linear Discriminant Analysis (LDA)** for **feature extraction** on a churn prediction dataset, and compare their impact on classification performance using logistic regression.

Data Preprocessing

- **One-Hot Encoding**:
  The categorical features gender and occupation were encoded using pd.get_dummies(), with drop_first=True to avoid multicollinearity. This created binary variables such as gender_Male, occupation_self_employed, and occupation_student.
- **Feature Selection**:
  A subset of features was selected for modeling. These included:
  - Demographic features: age, dependents, gender_Male, etc.
  - Account activity features: current_balance, previous_month_balance, etc.
- **Target Variable**:
  The target for prediction is churn — a binary indicator of whether a customer left the service.

Standardization

- **Why**: PCA and LDA are sensitive to the scale of input features.
- **How**: StandardScaler was used to normalize the feature matrix X, giving all features a mean of 0 and standard deviation of 1.

Principal Component Analysis (PCA)

- **Purpose**: To reduce the feature space while retaining as much variance in the data as possible.
- **Method**:
  - PCA was applied to the standardized data.
  - The cumulative explained variance was plotted to determine how many components are needed to capture ~90% of the total variance.
  - A new PCA model (PCA(n_components=0.90)) was fitted to retain only the necessary components (dimensionality reduction).
- **Visualization**:
  The first two principal components were plotted using a scatterplot, colored by churn. This helped assess class separability in reduced space.

Feature Extraction with PCA and LDA

Linear Discriminant Analysis (LDA)

- **Purpose**: Unlike PCA, LDA is supervised and aims to maximize class separation.
- **Method**:
    - LDA was applied to reduce the data to 1 dimension (n_components=1) because LDA with 2 classes allows only 1 discriminant axis.
- **Visualization**:
  A histogram was plotted showing the distribution of the single LDA component for each churn class.

Model Training and Evaluation

- **Logistic Regression** was applied to three sets of features:
    1. Original selected features (without dimensionality reduction)
    2. PCA-reduced features (90% variance retained)
    3. LDA-reduced feature (1D)
- **Train/Test Split**:
  The data was split into 70% training and 30% testing sets using a fixed random seed (42).
- **Evaluation Metrics**:
    - **Accuracy**: Measures the proportion of correct predictions.
    - **F1 Score**: Harmonic mean of precision and recall — more relevant for imbalanced classes like churn.

Model	Accuracy	F1 Score
Original	0.831	0.186
PCA (~90%)	0.818	0.006
LDA (1D)	0.821	0.065

Interpretation

- **Original Model** performed best in both accuracy and F1 score, likely due to using the full set of informative features.
- **PCA** slightly reduced accuracy and severely impacted F1 score. Since PCA is unsupervised, it may discard features relevant for churn classification.
- **LDA** retained class information better than PCA (due to supervision), but still underperformed compared to the original feature space.

SVM Modeling

1. Data Preparation and Scaling

- The dataset was prepared by selecting relevant features (`df_selected`) and splitting into features `X` and target `y` (`churn`).
- Data was split into training and testing sets using a 70/30 ratio with a fixed random seed for reproducibility.
- StandardScaler from sklearn was used to normalize the features. The scaler was fit on the training data and applied to both training and test sets.

2. SVM with Parameters

- An SVM with RBF kernel was trained using `C=10` and `gamma='scale'`, with class weight balanced to handle class imbalance.
- Model performance was evaluated using accuracy, precision, recall, and F1-score.
- The model was saved using pickle for future inference.

SVM with best parameters (C=10, gamma='scale')
Train Accuracy: 0.8173
Test Accuracy: 0.8182

3. Hyperparameter Tuning

- GridSearchCV was used to explore combinations of `C` and `gamma` values.
- The best model was selected based on F1-score using 3-fold cross-validation.
- The tuned model improved prediction performance and was evaluated on both training and testing data.

And the results were

Fitting 3 folds for each of 9 candidates, totalling 27 fits
Best Parameters: {'C': 10, 'gamma': 0.1}
Tuned Kernel SVM (RBF) with expanded parameters
Train Accuracy: 0.8173
Test Accuracy: 0.8117
Precision: 0.4851
Recall: 0.5775
F1-score: 0.5273

4. SVM with LDA

- Linear Discriminant Analysis (LDA) was applied to reduce dimensionality (1 component for binary classification).
- GridSearchCV was used on the LDA-transformed data to tune the SVM model.
- The best model was evaluated using training and test accuracy and F1 score.

Fitting 3 folds for each of 4 candidates, totalling 12 fits
Best Parameters: {'C': 1, 'gamma': 0.1}
SVM on LDA-transformed data with tuned parameters
Train Accuracy: 0.7017
Test Accuracy: 0.7031
F1 Score: 0.3882

5. SVM with PCA

- Principal Component Analysis (PCA) was used to retain 90% of the variance in the data.
- The SVM model with RBF kernel was trained on PCA-transformed data.
- Performance was measured using accuracy, precision, recall, and F1-score.

Kernel SVM (RBF) with PCA (~90%)
Train Accuracy: 0.8142
Test Accuracy: 0.8183
Precision: 0.5556
Recall: 0.0032
F1-score: 0.0064

6. Soft Margin SVM

- A soft margin SVM (C=1) was trained using the scaled data.
- This allows for some misclassification to improve generalization.
- The model was evaluated on training and test sets.

Soft SVM (RBF)
Train Accuracy: 0.8026
Test Accuracy: 0.8050
Precision: 0.4669
Recall: 0.5103

F1-score: 0.4877

7. Hard Margin SVM

- A hard margin SVM (C=1000) was used to strictly separate the classes.
- This model assumes no misclassification and may overfit if noise is present.

Hard SVM (RBF)
Train Accuracy: 0.8370
Test Accuracy: 0.8154
Precision: 0.4939
Recall: 0.6279
F1-score: 0.5529

8. Sample Predictions

- The final trained model and scaler were saved using pickle.
- Predictions were made on new input data using the saved model.
- For each record, the model predicted whether the customer would churn or not.

9. Execution in Streamlit

- Streamlit can be used to run an interactive frontend for churn prediction.
- The application can load the saved model and scaler and take user input to predict churn in real-time.