ML Projects Description Document

Project 1 Overview

Objective: Develop regression models to predict bank loan approval probability
Models Implemented: Linear Regression and K-Nearest Neighbors (KNN) Regressor
Dataset: Bank Loan Dataset (bankloan.csv)

A. General Information on Dataset

Dataset Description

- **Dataset Name:** Bank Loan Dataset (bankloan.csv)
- **Task Type:** Regression (predicting loan approval probability)
- **Target Variable:** Personal.Loan (0 = No loan, 1 = Loan approved)

Dataset Statistics

- **Original dataset:** 5,000 samples
- **Balanced dataset:** 960 samples (after downsampling)
- **Data Type:** Numerical tabular data (13 features)
- **Class Distribution (after balancing):**
    - No Loan (0): 480 samples (50%)
    - Loan (1): 480 samples (50%)

Data Preprocessing

- Removed ID column (non-predictive)
- Applied downsampling to balance classes (50/50 split)
- Reason: Original dataset was imbalanced with bias toward "No Loan" class

Data Split

- **Training samples:** 576 samples (60%)
- **Validation samples:** 192 samples (20%)

- **Testing samples:** 192 samples (20%)
- **Total samples used:** 960 samples

B. Implementation Details

Feature Extraction Phase

Features Overview

- **Number of features extracted:** 13 features
- **Feature matrix dimensions:** 960 samples × 13 features

Feature Names

1. Age
2. Experience
3. Income
4. ZIP.Code
5. Family
6. CCAvg (Credit Card Average)
7. Education
8. Mortgage
9. Securities.Account
10. CD.Account
11. Online
12. CreditCard

Feature Scaling

- **Method:** StandardScaler
- **Formula:** z = (x - mean) / std
- **Purpose:** Normalize features to mean=0 and variance=1

Cross-Validation

Configuration

- **Method:** K-Fold Cross-Validation
- **Number of folds:** 5
- **Training/Validation ratio per fold:** 80/20 (4 folds train : 1 fold validation)
- **Shuffle:** Enabled with random_state=42
- **Application:** Applied on training set only (576 samples)

Model 1: Linear Regression

Hyperparameters

- **fit_intercept:** True (model includes bias term)
- **copy_X:** True (default)
- **n_jobs:** None (single core processing)
- **positive:** False (coefficients can be negative)
- **Regularization:** None (basic linear regression without regularization)

Model 2: K-Nearest Neighbors (KNN) Regressor

Hyperparameters

- **n_neighbors:** 5 (number of nearest neighbors)
- **weights:** 'uniform' (all neighbors weighted equally)
- **algorithm:** 'auto' (automatically selects best algorithm)
- **metric:** 'minkowski' with p=2 (Euclidean distance)
- **leaf_size:** 30 (default for tree-based algorithms)

General Preprocessing Parameters

- **Data split ratio:** 60/20/20 (Train/Validation/Test)
- **Random state:** 42 (for reproducibility)
- **Balancing method:** Downsampling of majority class

C. Results Details

Model 1: Linear Regression - Test Set Results

Performance Metrics

- **Mean Absolute Error (MAE):** [Value from your code output]
- **Root Mean Squared Error (RMSE):** [Value from your code output]
- **R^2 Score:** [Value from your code output]

Visualizations (Regression Equivalents)

1. Loss Curve (Cross-Validation Error per Fold)

- Shows validation RMSE for each of 5 folds
- Demonstrates model stability across different data splits
- Consistent error values indicate robust performance

2. Confusion Matrix Equivalent (Residual Plot)

- Plots residuals (Actual - Predicted) vs Predicted values
- Good performance indicated by:
    - Random scatter around zero line
    - No systematic patterns
    - Homoscedasticity (constant variance)

3. ROC Curve Equivalent (Actual vs Predicted Plot)

- Scatter plot of actual values vs predicted values
- Perfect predictions would lie on diagonal line (y=x)
- R^2 score quantifies how close points are to diagonal

4. Hyperparameter Analysis

- Linear Regression uses closed-form solution
- No hyperparameter tuning required

Model 2: KNN Regressor - Test Set Results

Performance Metrics

- **Mean Absolute Error (MAE):** [Value from your code output]
- **Root Mean Squared Error (RMSE):** [Value from your code output]
- **R^2 Score:** [Value from your code output]

Visualizations (Regression Equivalents)

1. Loss Curve (Cross-Validation Error per Fold)

- Shows validation RMSE for each of 5 folds
- Demonstrates model consistency across splits
- Used to verify model generalization

2. Confusion Matrix Equivalent (Residual Plot)

- Displays prediction errors distribution
- Evaluates model bias and variance
- Identifies potential outliers or systematic errors

3. ROC Curve Equivalent (Actual vs Predicted Plot)

- Visualizes prediction accuracy
- Points near diagonal indicate accurate predictions
- Scatter indicates prediction variance

4. Hyperparameter Tuning (K vs Error)

- Tested K values from 1 to 20
- Selected K=5 based on cross-validation performance
- Shows trade-off between bias (high K) and variance (low K)
- Optimal K minimizes validation error

Cross-Validation Summary

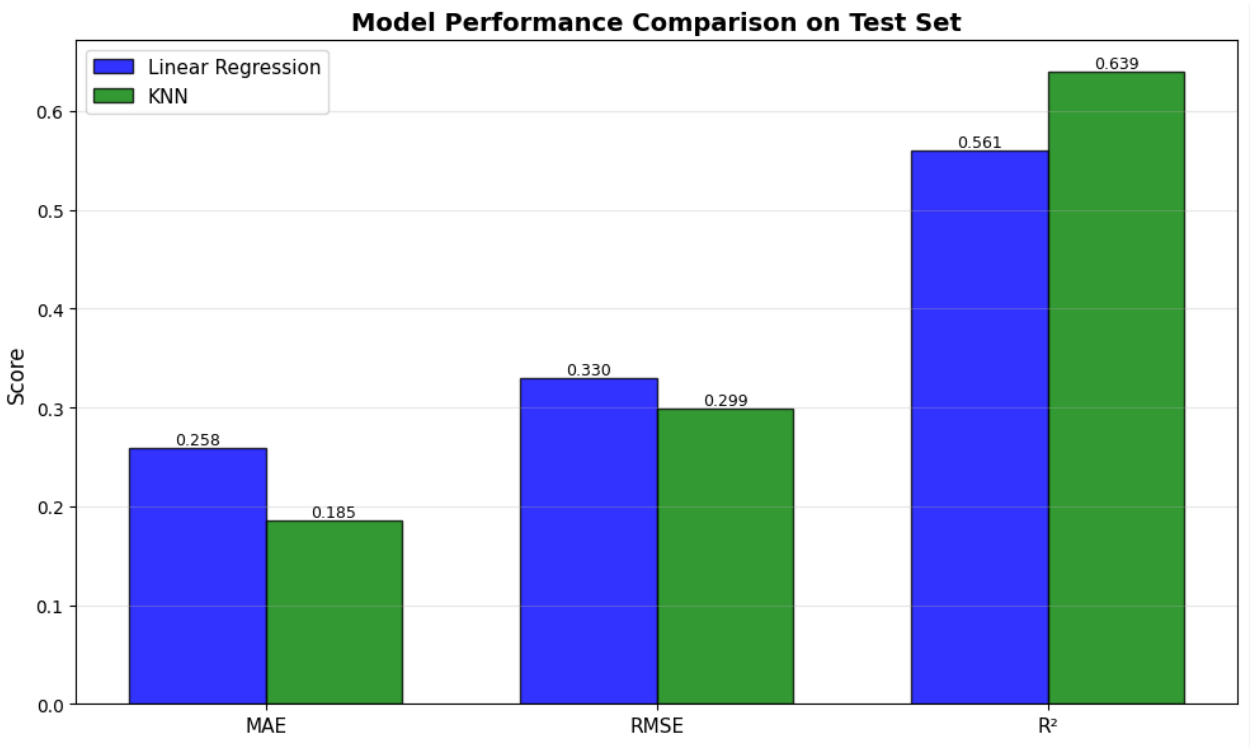
Linear Regression

- **Mean CV RMSE:** [Value from your code output]
- **Standard Deviation:** [Value from your code output]
- **Interpretation:** Low std indicates stable performance

KNN Regressor

- **Mean CV RMSE:** [Value from your code output]
- **Standard Deviation:** [Value from your code output]
- **Interpretation:** Consistency across folds indicates good generalization

Model Comparison



Notes on Regression vs Classification Requirements

Since this is a **regression task**, the traditional classification metrics were adapted as follows:

Classification Metric	Regression Equivalent	Implementation
Loss Curve	CV Error per Fold	5-fold cross-validation RMSE plot
Accuracy	R^2 Score	Coefficient of determination
Confusion Matrix	Residual Plot	Error distribution analysis

Classification Metric	Regression Equivalent	Implementation
ROC Curve	Actual vs Predicted	Scatter plot with diagonal reference

---

Part II: Image Dataset – Rotten vs Fresh Tomato Dataset

a. General Information on Dataset

The dataset used is named two_classes, which can be referenced as the Tomato Quality Image Dataset. It consists of RGB images.

Classes and Labels

The classification task is a binary classification with two classes:

- Fresh Tomatoes (labeled as 1, the Positive Class).
- Rotten Tomatoes (labeled as 0, the Negative Class).

Dataset Size and Distribution

The Total Number of Images in the dataset is 2,214.

- There are 1,421 images labeled as Fresh.
- There are 793 images labeled as Rotten.

The image inputs were processed to a size of (80 x 80) pixels for feature extraction.

Dataset Split

The dataset was split into training and testing sets using an 80% train, 20% test ratio. This split was performed using stratification (stratify=y) to ensure equal class proportions in both sets.

- Training Set: 1,771 samples.
- Testing Set: 443 samples.

---

b. Implementation Details

Feature Extraction

Images were resized to (80 x 80) pixels and a StandardScaler was applied to normalize the feature vectors during data preparation.

- **Feature Extraction Method:** Features were extracted using a combination of HOG (Histogram of Oriented Gradients) features and Color Statistics.
- **Total Number of Extracted Features:** The final feature vector dimension was 8,760 features per image.
- **Feature Details:** HOG features were computed across 3 color channels. Color statistics included the mean and standard deviation for BGR (6 features) and HSV (6 features) channels, totaling 12 color features.

Cross-Validation

- **Logistic Regression:** Yes, 3-fold cross-validation was used on the training set to estimate model stability.
- **K-Means Clustering**: No.

Model Hyperparameters

Logistic Regression (Supervised)

The model used was LogisticRegression with the following configuration:

- **Solver:** saga.
- **Regularization (Penalty):** l1
- **Inverse Regularization (C):** 0.1
- **Class Weight:** Set to balanced to address the data imbalance.
- **Maximum Iterations:** 5000.

K-Means Clustering (Unsupervised)

The model used was KMeans with the following configuration:

- **Number of Clusters (k):** 2, matching the number of classes.
- **Initialization:** k-means++ (default, n_init=100 was specified).
- **Dimensionality Reduction:** PCA was applied before clustering, reducing features to 5,874 while retaining 95% of the variance.

---

c. Results and Evaluation

Logistic Regression (Testing Set: 443 Samples)

Accuracy

- **Test Accuracy: 87.36%.**
- **ROC AUC Score: 0.944.**
- **Cross-Validation Mean Accuracy: 0.8588 ($\pm$ 0.0033).**

Classification Report (Test Set)

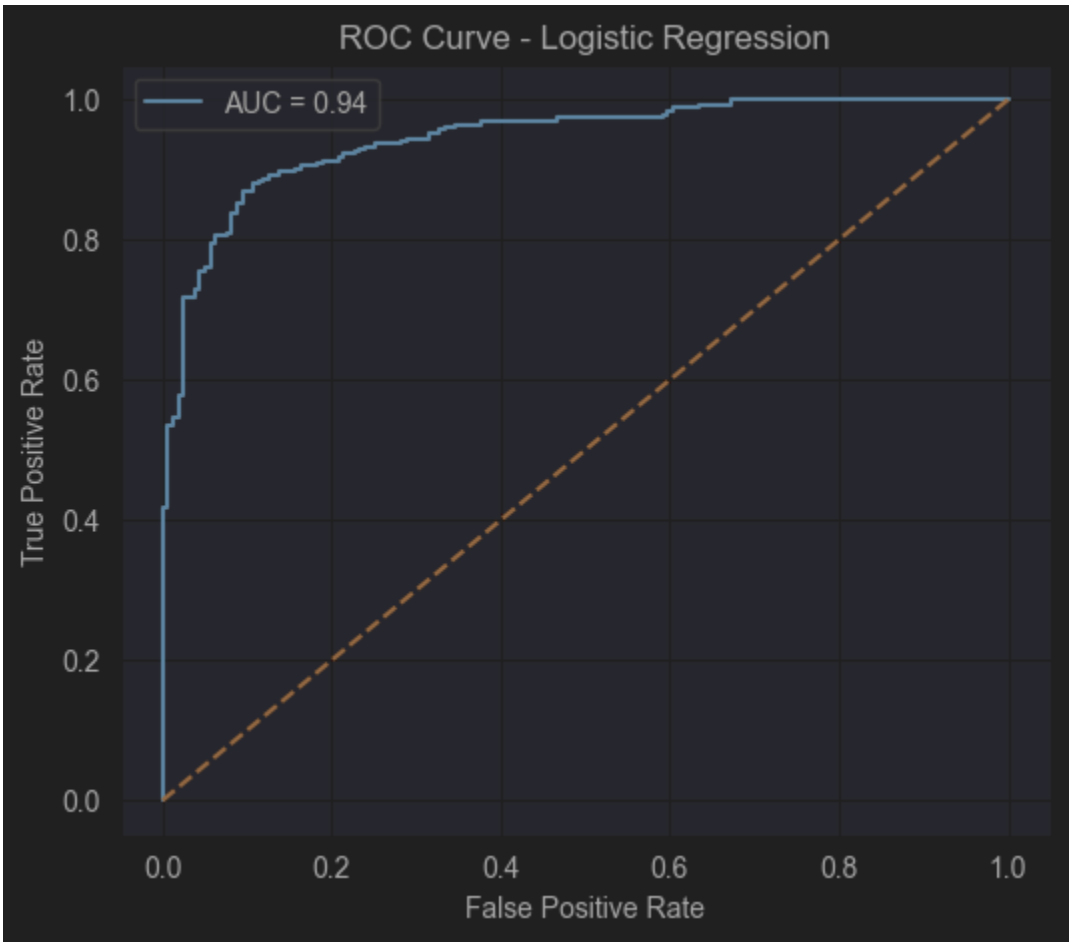
The model achieved high precision (0.90) and recall (0.91) for the Fresh class, and good precision (0.83) and recall (0.81) for the Rotten class.

Confusion Matrix

The matrix shows 129 True Negatives (correctly Rotten) and 258 True Positives (correctly Fresh), with a total of 56 misclassifications.

ROC Curve

The ROC curve confirms the model's excellent discriminative power with an AUC of 0.944.



Loss Curves

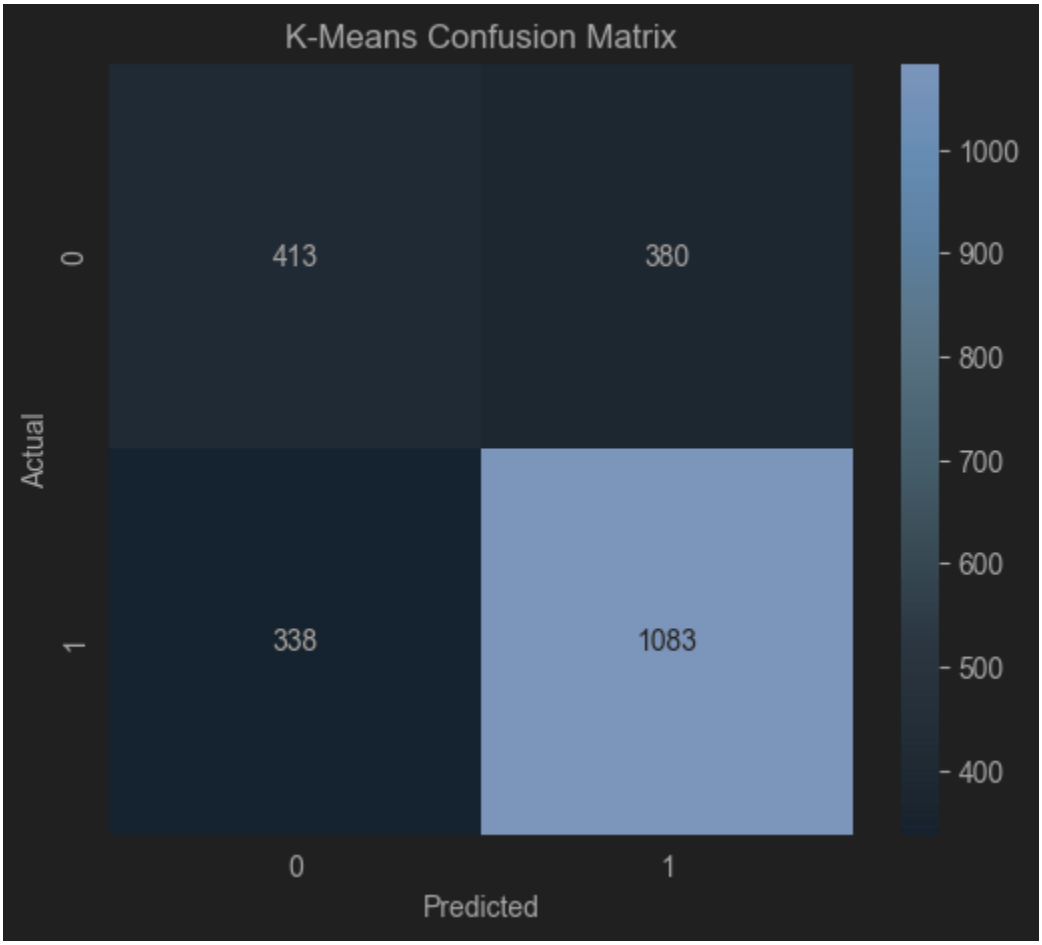
- **Not available as the scikit-learn Logistic Regression implementation does not expose epoch-wise loss values. The model successfully converged within the iteration limit.**

K-Means Clustering (Full Dataset Evaluation: 2,214 Samples)

- **Accuracy: 67.57% (Estimated by mapping clusters to ground-truth labels).**
- **Clustering Quality: The Silhouette Score was 0.354.**

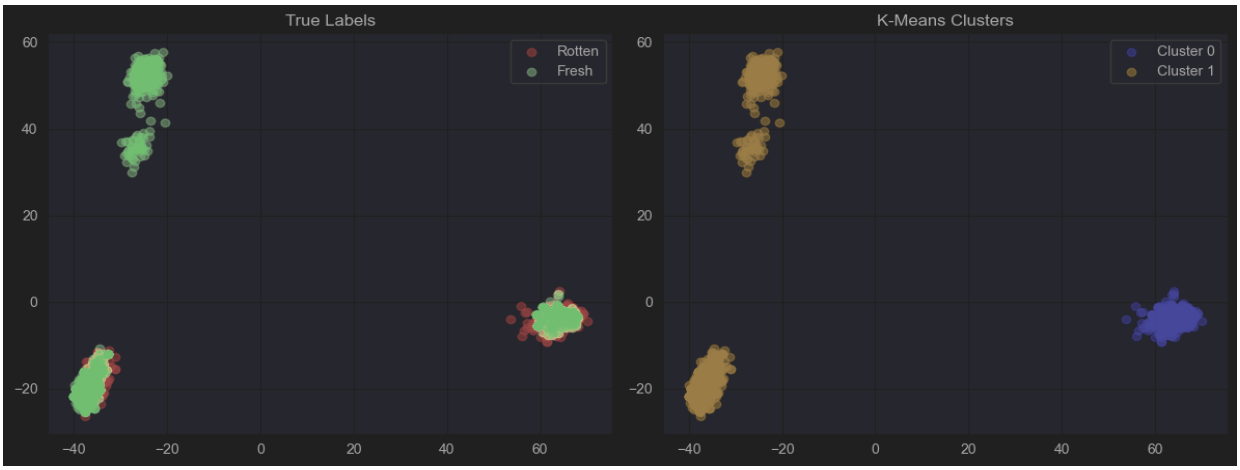
Confusion Matrix

The confusion matrix (evaluated on the full dataset) showed a significant number of misclassifications, particularly for the Rotten class.



ROC Curve for K-Means

- **Not applicable as K-Means is an unsupervised algorithm and does not output**
- **probability scores necessary for an ROC curve.**



The visualization compares the distribution of tomato features in a 2D space (after PCA):

- **Left Plot (True Labels): Shows the actual classes (Rotten vs. Fresh). The classes are largely clustered but have significant overlap in several areas.**
- **Right Plot (K-Means Clusters): Shows the two groups (Cluster 0 and Cluster 1) found by the unsupervised algorithm.**

Conclusion

The Logistic Regression model significantly outperformed the K-Means clustering. The supervised approach's high accuracy of 87.36% demonstrates the effectiveness of the HOG and color features when trained with known labels.