Python Project Report

Machine Learning Classification Using Python

Regression & Classification Modeling with Linear Regression, Decision Trees, and K-Nearest Neighbors

Leader
Frais Asghar

CMS ID
516900

Date
20 December 2024

Navigation

Table of Contents

Key Focus: Comprehensive analysis of machine learning algorithms with mathematical rigor and practical implementation insights

01 Introduction

Project Overview & Abstract

Project Objective

This project examines regression and classification modeling using an authentic dataset from the Kaggle Library. The study applies multiple machine learning algorithms to identify trends, establish decision boundaries, and evaluate classification accuracy.

Dataset Source

Kaggle Library
Authentic real-world dataset

Algorithms Implemented

Linear Regression
Regression prediction model

Decision Tree (DT)
Tree-based classifier

K-Nearest Neighbors (KNN)
Instance-based classifier

Key Performance Metrics

TP True Positive	**TN** True Negative
FP False Positive	**FN** False Negative

Evaluation Tools

- Confusion Matrix for classification outcomes
- Correlation Matrix for feature relationships
- Precision, Recall, and F1-Score metrics
- Decision boundary visualizations

02 Regression Modeling

Linear Regression: Theory & Mathematics

Algorithm Definition

Linear Regression is a fundamental supervised learning algorithm used to predict continuous values. It assumes a linear relationship between the dependent variable (output) and independent variables (input features).

Primary Objective

Fit a line that minimizes the error between predicted values and actual values

Mathematical Representation

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$$

- y Dependent variable (output)
- $x_1 \ldots x_n$ Independent variables (inputs)
- β_0 Intercept (y when all x = 0)
- $\beta_1 \ldots \beta_n$ Coefficients (feature weights)
- ε Residual error term (difference between actual and predicted y)

Regression Analysis Steps

1. **Data Splitting**
   Split dataset into train/test (e.g., 80:20)
2. **Model Training**
   Fit Linear Regression using training data
3. **Prediction**
   Predict outcomes for test data
4. **Evaluation**
   Calculate performance metrics

Key Characteristics

- ✓ Assumes linear relationships
- ✓ Predicts continuous values
- ✓ Minimizes sum of squared errors
- ✓ Highly interpretable model

02 Regression Modeling

Linear Regression: Performance Metrics

MAE
Mean Absolute Error

$$MAE = \frac{1}{n} \sum |y_i - \hat{y}_i|$$

Measures the average magnitude of errors in a set of predictions, without considering their direction. **Robust to outliers** as it uses absolute differences.

MSE
Mean Squared Error

$$MSE = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$$

Measures the average of the squares of the errors. **Penalizes larger errors** more heavily due to squaring, making it sensitive to outliers.

RMSE
Root Mean Squared Error

$$RMSE = \sqrt{MSE}$$

Square root of MSE, providing error in the **same units as the target variable**. More interpretable than MSE while maintaining sensitivity to large errors.

R^2
R-Squared (Coefficient of Determination)

$$R^2 = 1 - \left(\sum (y_i - \hat{y}_i)^2 / \sum (y_i - \bar{y})^2\right)$$

Represents the **proportion of variance explained** by the model. Ranges from 0 to 1, where 1 indicates perfect prediction.

Metric Selection: Choose MAE for robustness to outliers, MSE/RMSE for penalizing large errors, and R^2 for interpretable variance explanation.

03 Classification Modeling

Decision Tree Classifier: Theory & Concepts

Algorithm Overview

A Decision Tree is a supervised learning algorithm that uses a tree structure for classifying data or making predictions on continuous outcomes.

Core Mechanism

Iteratively splits dataset into sub-parts based on feature thresholds while maximizing purity at each node

Splitting Criteria

Gini Impurity

$$Gini = 1 - \Sigma p_i^2$$

Measures probability of misclassification

Entropy

$$Entropy = -\Sigma p_i \log_2(p_i)$$

Measures information uncertainty

Overfitting Prevention

Pruning techniques are used to constrain tree depth and prevent overfitting to training data.

- **Pre-pruning**: Stop tree growth early based on criteria
- **Post-pruning**: Remove branches after full growth

Decision Boundaries

DT creates axis-aligned boundaries that split feature space into separate class regions.

- ✓ Rectangular partitions of feature space
- ✓ Each split creates orthogonal divisions
- ✓ Easy to visualize and interpret

Advantages

✓ Easy & interpretable
✓ Handles all data types

Disadvantages

✗ Prone to overfitting
✗ Sensitive to noise

04 Classification Modeling

K-Nearest Neighbors: Theory & Implementation

Algorithm Definition

KNN is a non-parametric algorithm that classifies data points based on the majority class among their k nearest neighbors in the feature space.

Core Principle

Similar data points tend to exist in close proximity to each other in the feature space

Mathematical Foundation

1. Euclidean Distance

$$d(p, q) = \sqrt{\Sigma(p_i - q_i)^2}$$

2. Decision Rule

Assign the class with the **highest count** among the k nearest neighbors

Key Features

Effective for Small Datasets
Performs well with limited data

No Training Phase
Predictions computed at runtime

Non-Parametric
Makes no assumptions about data

Challenges

Parameter Tuning
Careful selection of k is critical

- → Small k: May lead to overfitting
- → Large k: May over-smooth boundaries

Best Practice: Use cross-validation to find optimal k value

05 Evaluation Metrics

Confusion Matrix & Classification Metrics

Confusion Matrix Structure

A summary table showing how well a classification model performs by comparing predicted vs. actual values.

	Predicted Positive	Predicted Negative
Actual Positive	TP True Positive	FN False Negative
Actual Negative	FP False Positive	TN True Negative

Accuracy

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Measures overall correctness—the proportion of all predictions that were correct.

Precision

$$\text{Precision} = TP / (TP + FP)$$

Measures exactness—the proportion of positive predictions that were actually correct. **High precision = low false positive rate**.

Recall (Sensitivity)

$$\text{Recall} = TP / (TP + FN)$$

Measures completeness—the proportion of actual positives correctly identified. **High recall = low false negative rate**.

F1-Score

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

Harmonic mean of precision and recall, providing a **balanced measure** when both metrics are important.

06 Feature Analysis

Correlation Matrix: Feature Relationship Analysis

Purpose & Definition

The correlation matrix provides information regarding relationships between features in a dataset, enabling identification of multicollinearity and feature importance.

- Identifies strongly correlated features
- Detects multicollinearity issues
- Determines feature importance
- Guides feature selection process

Correlation Coefficient Formula

$$r = \Sigma(x_i - \bar{x})(y_i - \bar{y}) / \sqrt{[\Sigma(x_i - \bar{x})^2 \times \Sigma(y_i - \bar{y})^2]}$$

Interpreting Correlation Values

+1.0 — **Perfect Positive**
Strong linear relationship (same direction)

0.0 — **No Correlation**
No linear relationship between variables

-1.0 — **Perfect Negative**
Strong inverse relationship

Practical Applications

1. **Feature Selection:** Remove highly correlated redundant features
2. **Multicollinearity Detection:** Identify problematic feature pairs
3. **Model Interpretation:** Understand feature relationships
4. **Data Exploration:** Discover hidden patterns

06 Visual Classification

Decision Boundaries: Visual Classification

What are Decision Boundaries?

Decision boundaries are graphical representations of how classifiers partition the feature space into distinct regions, each assigned to a specific class.

Key Insight

The shape and complexity of decision boundaries reveal how each algorithm learns and generalizes from training data

Decision Tree Boundaries

Axis-Aligned
Boundaries parallel to feature axes

Rectangular Regions
Feature space divided into boxes

Hierarchical Splits
Each split creates orthogonal divisions

- ✓ Easy to interpret and visualize
- ✓ May create complex boundaries with deep trees

KNN Boundaries

Non-Linear
Complex, irregular boundary shapes

Density-Dependent
Shape depends on data point distribution

k-Dependent Smoothness
Small k = jagged, Large k = smooth

- ✓ Highly flexible and adaptive
- ✓ Can model complex patterns

Comparison Summary

Characteristic	DT	KNN
Boundary Shape	Axis-aligned	Non-linear
Interpretability	High	Low
Flexibility	Moderate	High

07 Conclusion

Key Insights & Methodological Impact

Critical Success Factors

1. **Algorithmic Choice**
   Selecting the appropriate algorithm based on data characteristics and problem requirements is fundamental to success

2. **Parameter Tuning**
   Careful optimization of hyperparameters ensures optimal prediction modeling results and prevents overfitting

3. **Comprehensive Evaluation**
   Using multiple metrics provides a complete picture of model performance and reliability

Practical Utility

The methodologies demonstrated show hands-on utility in real-world machine learning applications across various domains including finance, healthcare, and predictive analytics.

Algorithm Summary

Linear Regression
Continuous value prediction with interpretable coefficients

Decision Tree
Hierarchical classification with clear decision rules

K-Nearest Neighbors
Instance-based learning with flexible boundaries

Key Achievements

- Comprehensive theoretical explanations with mathematical rigor
- Detailed performance metric analysis and interpretation
- Visual illustrations of decision boundaries for both classifiers
- Practical insights into algorithm selection and tuning

Thank You

Questions & Discussion

Leader

Frais Asghar

CMS ID

516900

Date

20 December 2024

Machine Learning Classification Using Python