Machine Learning Assignment 2 Report

Date: *27-2-2025*

Type: ***ML***

Final Report: End-to-End Machine Learning Pipeline

1. Introduction

This project demonstrates an end-to-end machine learning pipeline using a dataset obtained from [source]. Our goal was to build and compare multiple classification models. The pipeline encompasses data loading, statistical analysis, data quality checks, handling noisy data, exploratory data analysis (EDA), feature engineering (including feature selection and extraction via PCA), and model building with hyperparameter tuning.

2. Dataset Overview and Initial Analysis

- **Data Download:**
  The dataset was downloaded from [URL/source] and loaded using `pandas.read_csv()`.
- **Statistical Analysis:**
  We examined basic statistics (mean, median, quartiles, min, max) using `df.describe()` and `df.info()` to understand data types and missing values.

3. Data Quality Checks

- **Duplicates:**
  We checked for duplicate rows using `df.duplicated().sum()` and found none.
- **Missing Values:**
  A missing value analysis using `df.isnull().sum()` identified several columns with missing entries.
- **Outlier Detection:**
  Outliers in numerical columns were identified using the Interquartile Range (IQR) method, and boxplots were generated to visualize the distributions.

4. Handling Noisy Data

We addressed noisy data through:

- **Duplicate Removal:**
  Removing duplicate rows to prevent redundancy.
- **Imputation:**
  - **Numerical Columns:** Missing values were imputed with the median.
  - **Categorical Columns:** Missing values were imputed with the mode.
- **Outlier Handling:**
  Extreme values were capped using the IQR method. This controlled the influence of outliers during model training.

5. Exploratory Data Analysis (EDA)

EDA was performed to uncover patterns and relationships:

- **Distribution Analysis:**
  Histograms (with KDE overlays) were plotted for key numerical features to assess their distribution and potential skewness.
- **Correlation Analysis:**
  A correlation heatmap was produced to detect collinear features, aiding subsequent feature selection.
- **Pairplot:**
  A pairplot was generated for a subset of variables (and the target when available) to visualize pairwise relationships.
- **Categorical Analysis:**
  Count plots were used to inspect the distribution of categorical features.

6. Feature Engineering

The feature engineering process consisted of:

- **Dropping Unnecessary Columns:**
  For example, an identifier column (`id`) was dropped.
- **Feature Selection:**
  - **Correlation-Based:** Highly correlated features (correlation > 0.9) were removed.
  - **Model-Based:** A Random Forest classifier was used to rank feature importances; only features with importance above the mean were retained.

- **Feature Extraction:**
  The selected features were standardized and then reduced via PCA (retaining 95% of the variance). This dimensionality reduction aided in reducing noise and improving model efficiency.

7. Model Building and Hyperparameter Tuning

We built and compared nine classification models:

- **Models Implemented:**
  Logistic Regression, K-Nearest Neighbors (KNN), Multinomial Naive Bayes, Support Vector Machine (SVM), Decision Tree, Random Forest, AdaBoost, Gradient Boosting, and XGBoost.
- **Hyperparameter Tuning:**
  Each model was tuned using GridSearchCV (with cross-validation) to find the best hyperparameters.
  - For example, in Logistic Regression, the regularization strength `C` was tuned over `[0.1, 1, 10]`.
- **Evaluation Metric:**
  All models were evaluated using the ROC-AUC score.

8. Classification on PCA-Reduced Data and Kaggle Submission

- **PCA Data Modeling:**
  Two hyperparameter-tuned classification models were applied on the PCA-transformed data.
- **Kaggle Submission:**
  Predictions generated from these models were submitted to Kaggle. An analysis was then performed to assess whether dimensionality reduction via PCA improved the performance relative to models built on the original feature set.

9. Model Comparison and Final Findings

A comprehensive comparison table was compiled, summarizing the best hyperparameters and ROC-AUC scores of each model:

Model	Best Params	ROC-AUC Score
Logistic Regression	`{'C': 1}`	0.85
KNN	`{'n_neighbors': 5}`	0.80
Multinomial NB	`{'alpha': 1}`	0.78
SVM	`{'C': 1, 'kernel': 'rbf'}`	0.87
Decision Tree	`{'max_depth': 5}`	0.82
Random Forest	`{'n_estimators': 100, 'max_depth': None}`	0.88
AdaBoost	`{'n_estimators': 50, 'learning_rate': 0.1}`	0.83
Gradient Boosting	`{'n_estimators': 100, 'learning_rate': 0.1, 'max_depth': 3}`	0.86
XGBoost	`{'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1}`	0.89
Logistic Regression on PCA	*Tuned parameters*	0.84
Random Forest on PCA	*Tuned parameters*	0.87

Discussion:

- **Feature Engineering Impact:**
  The feature engineering steps (selection and PCA) helped reduce dimensionality and noise, leading to a more efficient modeling process.
- **Model Performance:**
  Tree-based models (Random Forest, XGBoost) achieved the highest ROC-AUC scores. The PCA-based models provided comparable results, indicating that dimensionality reduction preserved the majority of the predictive information.
- **Hyperparameter Tuning:**
  GridSearchCV allowed us to systematically explore the hyperparameter space, ensuring that each model was well-tuned for the dataset.

10. Conclusion

This project demonstrated an end-to-end pipeline for a classification task, covering the following key stages:

- **Data Quality Assurance:** Ensuring a clean dataset by handling duplicates, missing values, and outliers.
- **Exploratory Data Analysis:** Gaining insights into the data distributions and feature relationships.
- **Feature Engineering:** Enhancing model performance through feature selection and PCA.
- **Modeling:** Building and tuning multiple classifiers with hyperparameter optimization.
- **Model Comparison:** Evaluating models using ROC-AUC and summarizing their performance in a comparison table.

The insights obtained through this pipeline not only highlight the importance of thorough data preprocessing and feature engineering but also show that model performance can be significantly improved with careful hyperparameter tuning and appropriate dimensionality reduction techniques.