Lab 03

Classification Algorithms – Logistic Regression, Decision Trees, and Random Forests, KNN and SVM

Lab Objectives:

- Understand the concepts of classification algorithms.
- Implement Logistic Regression for binary classification.
- Explore Decision Trees for classification tasks.
- Apply Random Forests for ensemble-based classification.
- Evaluate and compare the performance of different classification models.

Learning Outcomes:

- Upon completing this lab, students should be able to:
- Understand the principles of classification algorithms.
- Implement Logistic Regression for binary classification.
- Utilize Decision Trees for classification tasks.
- Apply Random Forests as an ensemble learning method.
- Evaluate model performance using classification metrics.
- Compare and contrast the performance of different classification models

Classification Algorithms:

Classification is a type of supervised learning where the goal is to predict the categorical class labels of new instances based on past observations. Common classification algorithms include Logistic Regression, Decision Trees, and Random Forests.

here's a brief description of each classification algorithm:

Logistic Regression:

Logistic Regression is a statistical method used for binary classification problems (though it can be extended to multi-class classification).

- It predicts the probability that a given input belongs to a certain category.
- Despite its name, it's used for classification, not regression.
- It works by modeling the probability of the default class via the logistic function.

k-Nearest Neighbors (KNN):

- KNN is a non-parametric algorithm used for both classification and regression tasks.
- It's a simple and intuitive algorithm based on the idea of similarity: objects that are close to each other are more likely to be in the same class.
- In KNN classification, the class of a data point is determined by a majority vote of its k nearest neighbors.

Decision Trees (DT):

- Decision Trees are versatile supervised learning algorithms used for classification and regression tasks.
- They split the dataset into subsets based on the most significant attribute/feature.
- Each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (in classification) or a numerical value (in regression).

Random Forests:

- Random Forests is an ensemble learning method used for classification and regression tasks.
- It operates by constructing multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- Random Forests introduces randomness in two key ways: by bootstrapping the data (creating multiple subsets of the dataset) and by considering only a random subset of features when making splits in the decision trees.

Support Vector Machines (SVM):

- Support Vector Machines (SVM) is a powerful supervised learning algorithm used for classification and regression tasks.
- SVM is particularly effective in high-dimensional spaces and when the number of features exceeds the number of samples.

- In classification, SVM seeks to find the hyperplane that best separates the classes while maximizing the margin, which is the distance between the hyperplane and the closest data points (support vectors).
- SVM can handle linear and non-linear decision boundaries using different kernel functions (e.g., linear, polynomial, radial basis function).
- SVM aims to maximize the margin while minimizing the classification error, making it robust to overfitting.
- SVM is memory efficient because it uses only a subset of training points (support vectors) in the decision function.

Each of these algorithms has its own strengths and weaknesses, and the choice of algorithm depends on factors like the nature of the data, the size of the dataset, and the interpretability of the model.

Evaluation Metrics:

Common classification metrics include accuracy, precision, recall, F1-score, and ROC-AUC. These metrics provide insights into a model's overall performance and its ability to correctly classify instances.

Accuracy: Accuracy measures the proportion of correctly classified instances out of the total instances. It's a simple and intuitive metric but can be misleading in imbalanced datasets.

Precision: Precision measures the proportion of true positive predictions out of all positive predictions. It focuses on the accuracy of positive predictions and is useful when the cost of false positives is high.

Recall (Sensitivity): Recall measures the proportion of true positive predictions out of all actual positive instances. It focuses on capturing all positive instances and is useful when the cost of false negatives is high.

F1-Score: F1-score is the harmonic mean of precision and recall. It provides a balanced measure between precision and recall and is useful when there's an imbalance between classes.

Confusion Matrix: A confusion matrix is a table that summarizes the performance of a classification algorithm. It shows the number of true positives, true negatives, false positives, and false negatives.

Problem-Based Scenario:

You are working on a project to predict the survival of passengers on the Titanic. The dataset contains information about passengers such as age, gender, class, and whether they survived or not. Your goal is to build and compare different classification models to predict the survival of passengers based on their features. The real-world application of this project is to assist in making informed decisions related to passenger safety on future voyages.

The dataset can also be sourced using seaborn library.

Dataset Description:

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The training set should be used to build your machine learning models. For the training set, we provide the outcome (also known as the "ground truth") for each passenger. Your model will be based on "features" like passengers' gender and class. You can also use feature engineering to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

I have also included **gender_submission.csv**, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

Variables in the dataset:

pclass: A proxy for socio-economic status (SES)

1st = Upper

2nd = Middle

3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...

Sibling = brother, sister, stepbrother, stepsister

Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.

Lab Tasks:

In this lab, you have to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc).

Data Loading and Exploration:

- Load the training dataset (train.csv) using pandas and explore its structure.
- Check for missing values and handle them appropriately.
- Visualize distributions of numerical features (e.g., age) and categorical features (e.g., gender, class) to gain insights into the data.

Feature Engineering:

- Create new features that might be useful for predicting survival (e.g., family size combining sibsp and parch).
- Encode categorical variables (e.g., gender, class) using one-hot encoding or label encoding.

Model Building:

- Split the data into features (X) and target variable (y).
- Split the data into training and testing sets.

Implement the following classification algorithms:

- Random Forest (RF)
- k-Nearest Neighbors (KNN)
- Decision Trees (DT)
- Support Vector Machines (SVM)

Model Training and Evaluation:

- Train each model on the training data.
- Evaluate the performance of each model using appropriate metrics such as accuracy, precision, recall, and F1-score.

Model Comparison:

- Compare the performance of the different models.
- Visualize the results using confusion matrices or ROC curves.

Fine-tuning and Optimization (Optional):

- Perform hyperparameter tuning for each model to optimize their performance.

Prediction on Test Set:

- Once the best-performing model is identified, use it to make predictions on the test set (test.csv).