BAJAJ INSTITUTE OF TECHNOLOGY, WARDHA

Department of Computer Engineering

# Lab Manual

# Machine Learning Lab

(BTCOL606)

Year/Semester: 3rd Yr./VIth Sem

Prepared By

Mr. Vikas Palekar

Assistant Professor

Department of Computer Engineering

2024-25 (Even)

## EXPERIMENT LIST

Practical

Experiment 1: a) Creating and loading different datasets in Python.

b) Reshaping, Filtering, Scaling, Merging the data and
Handling the missing values in datasets.

Theory:

1. pandas

Pandas is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library.

2. numpy

Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices.

3. csv file

To use the dataset in our code, we usually put it into a CSV file. CSV stands for "Comma-Separated Values" files; it is a file format which allows us to save the tabular data, such as spreadsheets.

4. read_csv() & read_excel()

Now to import the dataset, we will use read_csv() & read_excel() functions of pandas library, which are used to read a csv & Excel file and performs various operations on it. Using this function, we can read a csv file locally as well as through an URL.

5. describe(), head(), tail(), shape

With .describe(), we can get an overview of the values each column contains. You can have a look at the first five rows with .head() whereas you can display the last five rows with .tail(). You also use the .shape attribute of the DataFrame to see its dimensionality.

6. Missing Value Imputation

The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

Some of the ways to handle missing data are:

By deleting the particular row: In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.

By calculating the mean: In this way, we will calculate the mean of that column which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

Implementation:

1. Using sklearn.impute.SimpleImputer:

Syntax:

```
class sklearn.impute.SimpleImputer(*, missing_values=nan, strategy='mean')
```

Univariate imputer for completing missing values with simple strategies. Replace missing values using a descriptive statistic (e.g. mean, median, or most frequent) along each column, or using a constant value.

2. sklearn.preprocessing.MinMaxScaler

Syntax:

```
class sklearn.preprocessing.MinMaxScaler(feature_range=(0, 1), ..)
```

Transform features by scaling each feature to a given range.

3. sklearn.preprocessing.StandardScaler

Syntax:

```
class sklearn.preprocessing.StandardScaler()
```

Standardize features by removing the mean and scaling to unit variance.

Conclusion: In this way, we understood the pre-processing and analysis steps required to be carried out before the actual start of the ML tasks.

Experiment 2: Study the use of metrics and scoring to quantify the quality of a model's prediction and plot the graphs

Theory:

1. matplotlib

Matplotlib is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code. There are five key plots that are used for data visualization. They are Bar graph (.bar()), Scatter plot (.scatter()), Histogram (.hist()), Area plot (.fill_between()), and Pie plot (.pie()). The plot() function is used to draw points (markers) in a diagram. By default, the plot() function draws a line from point to point.

Implementation:

1. sklearn.metrics.accuracy_score

Syntax:

```
sklearn.metrics.accuracy_score(y_true, y_pred)
```

Gives accuracy classification score.

2. sklearn.metrics.precision_score

Syntax:

```
sklearn.metrics.precision_score(y_true, y_pred, labels=None, pos_label=1)
```

Compute the precision.

3. sklearn.metrics.recall_score

Syntax:

```
sklearn.metrics.recall_score(y_true, y_pred, labels=None, pos_label=1)
```

Compute the recall.

4. sklearn.metrics.f1_score

Syntax:

```
sklearn.metrics.f1_score(y_true, y_pred, *, labels=None, pos_label=1)
```

Compute the F1 score, also known as balanced F-score or F-measure. 5. sklearn.metrics.confusion_matrix

Syntax:

```
sklearn.metrics.confusion_matrix(y_true, y_pred, labels=None)
```

Compute confusion matrix to evaluate the accuracy of a classification.

Conclusion: In this way, we understood the metrics to evaluate the performance of the trained model. We also studied the matplotlib library to visualize the data.

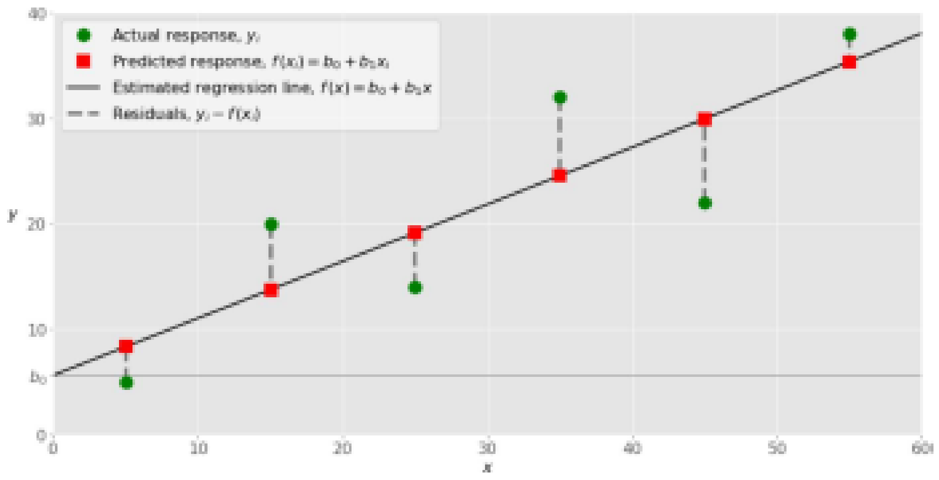
Experiment 3: Implement linear regression in Python on a given dataset.

<u>Theory:</u>

1. Linear Regression

Regression is a technique from statistics that is used to predict values of a desired target quantity when the target quantity is continuous. When implementing linear regression of some dependent variable �� on the set of independent variables x= ($x_1$,…,$x_r$), where r is the number of predictors, you assume a linear relationship between y and x: y= $\beta_0$ + $\beta_1 x_1$ + ……+ $\beta_r x_r$+ ϵ. This equation is the regression equation. $\beta_0$, ��$_1$,…, $\beta_r$ are the regression coefficients, and �� is the random error.

The following figure illustrates simple linear regression:



<u>Implementation:</u>

1. sklearn.linear_model.LinearRegression

Syntax:

```
class sklearn.linear_model.LinearRegression()
```

Gives ordinary least squares Linear Regression.

2. corr() - Compute pairwise correlation of columns, excluding NA/null values.

3. reg.coef_ - Estimated coefficients for the linear regression problem. 4. reg.intercept_ - Independent term in the linear model.
5. sklearn.impute.KNNImputer

Syntax

```
class sklearn.impute.KNNImputer(*, missing_values=nan, n_neighbors=5, weights='
uniform')
```

Imputation for completing missing values using k-Nearest Neighbors.

6. sklearn.model_selection.train_test_split

Syntax:

```
sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, r
andom_state=None, shuffle=True) _
```

Split arrays or matrices into random train and test subsets.

7. fit()-Fit linear model & predict()-Predict using the linear model.  8. sklearn.metrics.r2_score

Syntax:

```
sklearn.metrics.r2_score(y_true, y_pred) _
```

Computes R2 (coefficient of determination) regression score function.

<u>About Dataset:</u>

(Describe your dataset)

Conclusion: In this way, we have studied and implemented the linear  regression algorithm and predicted the values of continuous valued attribute.

## Experiment 4: Build a decision tree model for a given dataset

<u>Theory:</u>

1. Decision Tree

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value. Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

<u>Implementation:</u>

1. sklearn.preprocessing.LabelEncoder

Syntax:

```
class sklearn.preprocessing.LabelEncoder()
```

Encode target labels with value between 0 and n_classes-1.

2. sklearn.tree.DecisionTreeClassifier

Syntax:

```
class sklearn.tree.DecisionTreeClassifier(criterion='entropy', )
```

Trains decision tree classifier.

3. sklearn.model_selection.cross_val_score

Syntax:

```
sklearn.model_selection.cross_val_score(estimator, X, y=None, *, groups=None, scoring=None, cv=None, n_jobs=None, verbose=0, fit_params=None, pre_dispatch='2*n_jobs', error_score=nan)
```

Evaluate a score by cross-validation.

<u>About Dataset:</u>

(Describe your dataset)

Conclusion: In this way, we have studied and implemented the Decision Tree Algorithm for the classification task. We also performed the k-fold cross validation.

# Experiment 5: Implement Naïve Bayes Classifier using sklearn library

Theory:

1. Naïve Bayes Classifier

   Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features in machine learning.

   Problem statement:

   – Given features $X_1, X_2, \ldots, X_n$

   – Predict a label Y

$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

or

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

- P(H) is the probability of hypothesis H being true. This is known as the prior probability.
- P(E) is the probability of the evidence(regardless of the hypothesis).
- P(E|H) is the probability of the evidence given that hypothesis is true.
- P(H|E) is the probability of the hypothesis given that the evidence is there.

Implementation:

1. sklearn.naive_bayes.GaussianNB

Syntax:

```
class sklearn.naive_bayes.GaussianNB()
```

Creates Gaussian Naive Bayes (GaussianNB) classifier.

About Dataset:

(Describe your dataset)

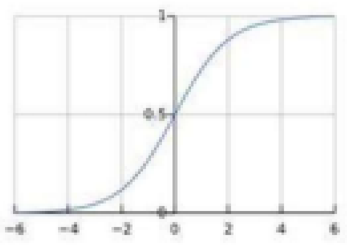
Conclusion: In this way, we implemented the Gaussian Naïve Bayes classifier on a given dataset and measured its performance.

## Experiment 6: Write a Python program to implement Logistic Regression and plot the graphs.

Theory:

1. Logistic Regression

It's a classification algorithm that is used where the target variable is of categorical nature. The main objective behind Logistic Regression is to determine the relationship between features and the probability of a particular outcome.



Logistic regression - Logit function

Implementation:

1. sklearn.linear_model.LogisticRegression

Syntax:

```
class sklearn.linear_model.LogisticRegression(multi_class='auto')
```

Creates Logistic Regression (aka logit, MaxEnt) classifier. In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme

About Dataset:

(Describe your dataset)

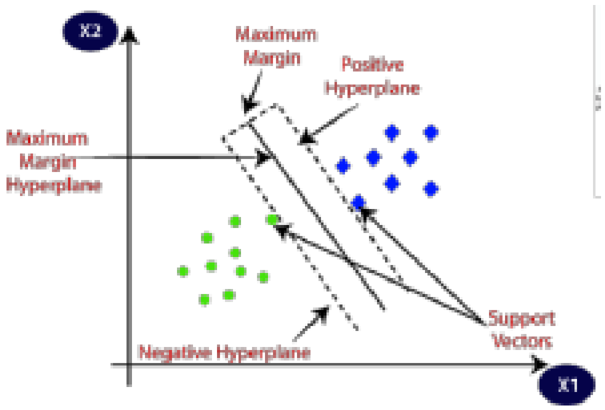
Conclusion: In this way, we understood the working of the logistic regression algorithm for classification task. We also plotted the sigmoid function using the probabilities.

Experiment 7: Build Support Vector Machine (SVM) model

<u>Theory:</u>

1. Support Vector Machine (SVM)

   Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.



<u>Implementation:</u>

1. sklearn.svm.SVC

Syntax:

```
class sklearn.svm.SVC(kernel='rbf', degree=3)
```

Support Vector Classification.

<u>About Dataset:</u>

(Describe your dataset)

Conclusion: In this way, we understood and implemented the SVM algorithm and tried to plot the maximum separating hyperplane using linear classifier.

Experiment 8: Train the Neural Network on a given datatset

Theory:

1. Neural Network

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

2. CNN

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

a) Convolutional layer
b) Pooling layer
c) Fully-connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

3. Keras

Keras is the recommended library for deep learning in Python. Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Well, Keras is an optimal choice for deep learning applications.

4. Tensorflow

TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

About Dataset:

(Describe MNIST dataset)

Conclusion: In this way, we trained the convolutional neural network to recognize the hand-written digit using the Keras and Tensorflow libraries.

Experiment 9: Apply Ensemble learning and evaluate the prediction

Theory:

1. Ensemble learning

   Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

Implementation:

1. sklearn.ensemble.RandomForestClassifier

   class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, ..)

   Creates a random forest classifier.

About Dataset:

Dataset - Rice type classification

This is a set of data created for rice classification. In the class column, Jasmine type is recorded as 1, whereas Gonen type as 0.

Conclusion: In this way, we understood the rationale behind the ensemble learning and implemented the Random Forest algorithm on the rice classification dataset.

## Experiment 10: Implement K-means Clustering algorithm and evaluate its performance

Theory:

1. K-means Clustering Algorithm

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some pre designated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated for example by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

K-Means Algorithm: Steps

1. Load data set
2. Clusters the data into k groups where k is predefined.
3. Select k points at random as cluster centers.
4. Assign objects to their closest cluster center according to the Euclidean distance function.
5. Calculate the centroid or mean of all objects in each cluster. 6. Repeat steps 3, 4 and 5 until the same points are assigned to each cluster in consecutive rounds.

Implementation:

1. sklearn.cluster.KMeans

Syntax:

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k
means++', n_init='warn', max_iter=300, tol=0.0001, verbose=0, random_state=None,
copy_x=True, algorithm='lloyd')
```

Implements K-Means clustering.

2. kmeans.labels_ - Labels of each point
3. kmeans.cluster_centers_ - Coordinates of cluster centers
4. kmeans.inertia_ - Sum of squared distances of samples to their closest cluster center, weighted by the sample weights if provided.

About Dataset:

(Describe your dataset)

Conclusion: In this way, we implemented the K-Means clustering algorithm and visualized the clusters.

******************************