Machine Learning Lab : Experiment no. 1

Python Libraries For Datascience

Aim : Python Libraries for Data science
a) Pandas b) Numpy c) scikit-learn d) Matplotlib

Theory :

1) Pandas is a library in python used for working with datasets. It has functions for analyzing, cleaning, exploring and manipulating data.

This library includes —

(a) read

The read functions are used to load data from various file formats into a pandas Dataframe. The most commonly used is 'read_csv' which reads a comma-seperated values in file. eg. df = pd.read_csv ('file.csv')

(b) head ()

The head () function returns first n rows of a Dataframe. It is useful for quickly inspecting the beginning of a Dataframe to understand data and structure.
eg. df.head () or df.head (10)

(c) tail ()

Similar to head (), the tail () function returns the last n rows of a data frame. It is useful for checking the end of dataset especially when dataset is large.
eg. df.tail () or df.tail (10).

(d) shape
The shape attribute returns a tuple representing the no. of rows & columns in a DataFrame. It's not a function but an attribute, meaning it doesn't require parentheses.
eg. df.shape

(e) info
The info() function provides a concise summary of dataframe. This is useful for getting the quick overview of the Data Frame's structure & data types.
eg. df.info.

(f) size
This attribute returns total no. of elements in DataFrame. It is calculated as the no. of rows multiplied by the no. of columns.
eg. df.size

(g) isna()
This function is used to detect missing values in a Data Frame. It returns a data frame of the same shape where 'True' indicates missing value & 'False' indicates non-missing value.
eg. df.isna()

(h) describe()
This generates descriptive statistics for numerical columns in Dataframe, which is helpful for getting a quick statistical summary of the data.
eg. df.describe()

(i) nunique()

This function returns the no. of unique values in each column or a specified axis. This is useful for understanding the diversity of values in a column, especially in a categorical data.

eg. df.nunique()

2) Numpy is a python library which is useful for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform and matrices.

This library includes —

(a) min() & max()

The min() function returns the minimum value of an array along a specified axis. Similarly, the max() function returns the maximum value of an array along a specified axis or the maximum value of the flattened array.

eg. np.min(array), np.max(array)

(b) mean()

The mean() function calculates the arithmetic mean of the elements in an array along a specified axis. If no axis is specified, it calculates the mean of the flattened array.

eg. np.mean(array)

(c) std()

This function calculates the standard deviation of the elements in an array along a specified axis. It measure

the amount of variation or dispersion of a set of values.
eg. np.std(array)

(d) median()
This function returns the median of elements in an array along a specified axis. This is the value that separates higher half from the lower half of the data.
eg. np.median(array)

(e) percentile()
This function computes the q^{th} percentile of the elements in an array along a specified axis. It is a measure that indicates the value below which a given percentage of observations in a group.
eg. np.percentile(array, q) where q is percentile value.

(f) stack()
This function joins a sequence of arrays along a new axis. It is used to concatenate arrays which introduces a new dimension.
eg. np.stack(array1, array2)

(g) vstack()
This function stacks arrays in a sequence vertically, which is equivalent to concatenating along axis 1 after arrays have been horizontally stacked.
eg. np.vstack(array1, array2)

(h) hstack()
This function stacks the elements of an arrays in a sequence horizontally. It is equivalent to concatenating

along axis1 after the arrays have been horizontally stacked.
eg. np.hstack (array1, array2)

(g) sort()
This function sorts the elements of an array along a specified axis. If no axis is specified, it sorts the array in an ascending order.
eg. np.sort(array).

3) Matplotlib is a comprehensive library for creating a static, animated and interactive visualizations in Python. It makes both hard and easy things possible. Matplotlib is primarily used for 2D plotting where the submodule 'pyplot' is most commonly used which provides functions to create figures, plots, bar charts, line, scatter plots, etc.

— Bar chart
It is used to represent the categorical data with rectangular bars. The length of each bar is proportional to the valve it presents.
— Line chart
It is used to display data points over a continuous period of time, connected by straight lines, useful for showing trends over time.
— Scatter plot
It is used to plot represent the relationship between the two continuous variables. It is useful for identifying correlations between variables.

common usage :-

```
import matplotlib.pyplot as plt
x = [1,2,3,4]
y = [5,7,9,11]
plt.plot(x,y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Line Plot Example')
plt.show()
```

Example :

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('file.path')
df.plot()
df.show()
```

- 'pandas' is imported as 'pd' is a library used for data manipulation & analysis. 'matplotlib.pyplot' is a library for creating interactive visualizations in python
- A file path is reads using 'read_csv' where it is then loaded into a pandas Dataframe named 'df'.
- A Basic plot of data in a dataframe is determined by plot() function which will plot each column against index. Finally, 'plt.show()' is explictly called to display the plot.

4) scikit-learn (often abbreviated as 'sk-learn') is a popular Python Library used for machine learning. It is built on the top of other python libraries like Numpy, SciPy, matplot & is widely used for implementing & experimenting with various machine learning algorithms.

This can be understood by —

(i) importing necessary libraries

```
from sklearn.model_selection
import train_test_split
from sklearn.neighbors
import kNeighborsClassifier

X = [18, 22, 25, 30, 35]
y = [1, 1, 1, 0, 0]

X_train, X_test, y_train, y_test =
        train_test_split(X, y, test_size=0.3, random_state=42)
model = kNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(predictions)
```

(ii) train_test_split helps split our data into training & testing sets.

- 'kNeighborsClassifier' is a simple machine learning model.
- X is list of ages & y is list whether true or false.
- Data is splitted into 70% of training & 30% of testing.

we create 'kNeighborsclassifier' and train it using training data. model then makes predictions on the test guess, guessing whether the age & prediction is True or False on depending negatives or positives respectively.

Conclusion : In this experiment, we studied various Machine Learning libraries & also implemented the libraries. These included Numpy, pandas, scikit Learn and Matplotlib.

11-24.

Experiment no. 02
Evaluation Metrics

Aim : Evaluation Metrics :-
a) Accuracy b) Precision c) Recall d) F1-score.

Theory :

The accuracy of a classifier is calculated as the ratio of the total no. of correctly predicted examples by the total no. of samples.

$$\text{Accuracy} = \frac{\text{Total no. of correctly produced samples}}{\text{Total no. of samples}}$$

Confusion matrix

A confusion matrix is a N-dimensional square matrix, where N represents total no. of target classes or categories. confusion matrix can be used to evaluated a classifier whenever the data set is imbalanced. Let us consider a binary classification problem i.e the no. of target classes are 2. A typical confusion matrix with two target classes.

		Predicted value No	Predicted value Yes
Actual value	No	TN (True Negative)	FP (False Positive)
Actual value	Yes	FN (False Negative)	TP (True Positive)

Precision :- (or positive predicted value)
Precision is the ratio of true positives (TP) by sum of true positives (TP) and False positives (FP).

$$\text{Precision} : \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Recall :-
Recall is concerned with the quantity of the relevant instances captured by the model. A high recall indicates that the model has a low rate of false negatives.
Recall can be calculated using the formula :-

$$\text{Recall} : \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

F1 score :-
It is calculated as the harmonic mean of precision and recall. It is important evaluation metric that is commonly used in classification tasks to evaluate performance of a model.
F1 score can be calculated using the formula :-

$$\text{F1 score} : \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4 terms to remember while making confusion matrix :-

1) True Positive.
It is the case where the predicted & both the actual value is True.

2) True Negative
It is the case where the predicted and the actual value is False.

3) False Positive
It is the case where the predicted value is True but Actual value is False.

4) False Negative
It is the case where predicted value is False but the actual value is True.

Conclusion :
By understanding & using confusion matrix, we effectively assess the performance of machine learning model. By using accuracy, precision, recall & F1-score as evaluation metrics, we determine the performance of the model.

Experiment no: 03.

Aim: Train and Test Sets of splitting learn and Test Data.

Theory:

(i) Data splitting in machine learning.
Data splitting is an eventual process in machine learning that ensures models are trained, validated and tested properly. The aim is to build a model that can generalize well to unseen data and avoid issues such as overfitting or underfitting. This approach simulates how the model will performance of the model after the training is complete.

(ii) Training Data
Training Dataset is a subset of the entire dataset that the machine learning algorithm uses to learn the relationships between input features (independent variables) and the output labels (dependent variables). This subset of data typically consists of 60 to 80% of the total data, depending on the size of the dataset.

(iii) Validation Data:
Validation Data is a seperate subset from the training data, typically around 10% to 20% of the total data. Its primary purpose is to fine-tune

model parameters and select the best model configuration. While the model does not learn from the data, it helps guide decisions on which model to choose based on performance metrics.

(iv) Testing Data

Testing Data is the final hold-out set that the model has never seen during training or validation. The subset usually makes up around 10 to 20% of the total data. The test dataset provides an unbiased evaluation of a models performance after it has been trained. Metrics such as accuracy, precision, recall, F1 score or mean absolute error (MAE) are used to evaluate the performance on the test data.

(v) Parameters of Data Splitting

To achieve a well-structured split, the following parameters are important.

1. Test size

The test size parameter specifies the proportion of the dataset to be allocated to the test data.

For instance, if test-size = 0.2 i.e 20% of the dataset will be used for testing and the remaining 80% will be used for training.

Choose the right test size is crucial. A too small test set may not provide an accurate estimate of model performance, while a too large set may leave insufficient data for training.

2. Train size

The train-size parameter controls the proportion of data allocated to the training set.

If this is not specified, the remaining data (after setting the test size) is automatically assigned to the training data set.

Ideally, you want to leave enough data from training set to allow the model to learn effectively, but also ensure there's enough data for reliable validation and testing.

3. Random state

The random state parameter controls the randomness of the data shuffling before the actual splitting.

This is crucial for reproducibility, while setting a fixed random state ensures that every time, you run the code, the data is split in the same way.

Train-test_split() function

The train_test_split() function from the sklearn.model selection module is a utility that splits the dataset into random training, and testing subsets. It shuffles the dataset and returns a randomly selected subset for training and another for testing based on the parameters specified.

Conclusion :

Through the process of dividing our dataset into training and testing sets, we effectively evaluated the performance of our machine learning model. By training the model on training set, we are able to gain insights into model's ability to predict on an unseen data.

Experiment no. 04

Aim : Linear Regression

Theory :

1) What is Linear Regression?

Linear Regression models the relationship between a dependent variable (outputs) and the set of one or more variables (inputs) by fitting a straight line.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots \beta_x n_x$$

where,

- Y is the predicted output.
- $x_1, x_2 \ldots x_n$ are input features.
- β_0 is the intercept.
- $\beta_1, \beta_2 \ldots \beta_n$ are the coefficients.

2) Steps in a Linear Regression Experiment.

a. Data collection and preprocessing.
b. Train test split
c. Model training
d. Model Evaluation
e. Interpretation

3) Key concepts.

- Cost function : measures the error between the actual and the predicted values.

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

- Gradient Descent : A common optimisation technique used to minimise the cost function and find the best fit.

- Evaluation Metrics : Common metrics include R^2 (explained variance) and RMSE (error magnitude).

- Regularization (optional)
  If your model overfits, consider ridge (R^2) regularization to penalize large data coefficients and improve generalization.

What is Best Fit Line?

Our primary objective while using linear regression is to locate best-fit-line, which implies that the error between the actual & predicted values should be kept to minimum.
  Its equation provides a straight line that represents relationship between the dependent and independent variables.

Conclusion :
  Linear Regression predicts continuous outcomes by relationship between inputs and outputs with a straight line.

Experiment no. 5

Aim: Multivariable Regression

Theory:

What is a multivariate regression?
It is a statistical tool or a technique that uses a mathematical model to analyse the relationship between multiple independent variables and a dependent variable. It is one of the simplest Machine Learning Algorithm that comes under the class of supervised learning Algorithms i.e when we are provided with a training dataset.

The solution for solving problems using multivariable regression is divided into various parts as follows.

- Selecting features
  : Finding the features on which a response variable depends is one of the most important steps in multivariable regression.

- Normalizing features
  The features are then scaled in order to bring in a range (0,1) to make better analysis by changing the value of each feature by —

$$X_i = \frac{x_i - \mu_i}{\sigma_i}$$

where,
x_i = training examples for ith feature

μ_i => mean of ith feature.
s_i => range of ith feature.

- Selecting Hypothesis and cost function
  A hypothesis is a predicted value of respons variable represented by $h(x)$.
  Cost function defines the cost for wrongly predicting hypothesis.

- Minimising the cost function.
  Cost minimisation algorithm runs over the datasets which adjusts the parameters of hypothesi Gradient descent algorithm is a good choice for minimising the cost function in case of multivariab regression.

- Testing the hypothesis
  The hypothesis function is then tested to over the test set to check its correctness & efficiency.

Implementation:
Multivariate regression technique can be implemented efficiently with help of matrix operations with the use of numpy library containing the definitions & operations.

conclusion:
Thus, it is used to evaluate predictors for the continuous distributed outcome variables, while computing coefficient.

Experiment no.6

Aim : Decision Tree Algorithm Implementations.

Theory:

(i) Decision Tree Basic structure

A decision tree has a hierarchical structure with a root node, branches, internal nodes and leaf nodes.

Decision Trees helps formalize brainstorming process & make complex decisions.

Root node

A decision tree consist of a root node which is the topmost node in tree structure, has no parent node but one or more child nodes.

A decision tree starts with a root node that signifies whole population, which then seperates into two or more uniform groups with a method called splitting. These undergoes further division which is identified as decision nodes, while the ones that don't divide are terminal nodes or leaves.

A segment of complete tree is referred to as a branch.

ASM (Attribute selection Measure)

It is a technique that selects best attribute or feature to split a data partition into individual classes. The goal is to split data into partitions, such as all the tuples belong to the same class.

Here are some popular selection of attribute measures:-
—Information gain & Gini Index
:- popular attribute selection method.
—Gain Ratio:
A measure that depends both onto information gain and the no. of outcomes of a feature.

—Gini Index
It is a measure of how impure or mixed dataset is & often used in decision Trees to classify data. It measures probability of misclassifying an item randomly selected from a dataset.

Advantages of Decision Tree
—Data Types: Decision Trees can handle both numerical & categorical data.
—missing values: It can handle missing values in dataset.
—model validation: It can be validated using statistical tests.

Disadvantages of Decision Tree
—overfitting: They can be prone to it.
—Bias: They can be biased if they fit too closely in data.
—sensitivity: small changes in data can lead to big differences in outcomes.
—complexity: They can be more complex.

Conclusion:- Decision trees assist analysts in upcoming choices. The tree creates a visual representation of all possible outcomes, rewards & follow up decisions in one document.

Experiment no: 7

Aim: Random Forest Algorithm implementation.

Theory:

(i) Random forest Algorithm

— The random forest algorithm is a machine learning technique that uses multiple decision trees to make predictions.
— The algorithm trains many decision trees on different subsets of a dataset, then uses majority voting to select the best prediction.

(ii) Random forest algorithm structure

— The random forest algorithm is a machine learning algorithm that uses multiple decision trees to make the predictions. The algorithm's structure and working can be summarized as follows:—

— Splitting the ~~features~~ dataset: The data set is divided into the subsets, either by randomly choosing features to train each tree.
— Training decision trees: Decision trees are trained on the subsets.
— Aggregating results: The results from each tree are combined to produce a more accurate prediction. For the classification, the most voted prediction is selected, while for regression, the average of the outputs of all trees is chosen.
— Validating the model: The model is validated.

(iii) Random Forest working

- It grows multiple decision trees which are merged together for more accurate prediction. The logic behind the Random Forest model is that multiple uncorrelated models perform much better as group than they do alone.
- When using Random forest for classification, each tree gives a classification or a 'vote'. The forest chooses the classification with majority of 'votes'. When using the Random Forest for regression, the forest picks average of outputs of all trees.

(iv) Assumptions for Random forest.

— Assumptions includes —

1) There should be actual values in the feature variable; this leads to prediction of more accurate results rather than a random guess by the algorithm.
2) The data is normally distributed.
3) The prediction from individual trees must have very low correlations.
4) The input data is continuous and the target data is discrete.

(v) Advantages of Random Forest

— Advantages includes :-

1) A reduced risk of overfitting which is attributed to multiple decision trees involved in its prediction functi
2) This algorithm unlike many other types, is very flexible in that it can perform both the classificati

and the regression tasks.

Applications :-

1) Financial sector : A range of models aimed at detecting fraudulent transaction & other activities of cyber criminals has leveraged on Random Forest.
2) E-commerce : It could be applied into it to understand the behaviour of customer & push the growth of business.
3) Health care sector : It's application lies in the prediction of diseases like cancer and heart diseases.

Conclusion :

The Random Forest algorithm is a supervised machine learning algorithm that uses a collection of decision trees to solve classification & regression problems. It's a powerful method that is effective for a wide range of data types including binary, continuous & categorical.

Experiment no. 08

Aim : Naive Bayes Classification Algorithm Implementation.

Theory :

Naive's Bayes Classification Algorithm.

— It is a supervised learning algorithm which is based on Bayes theorem and for solving the classification problems.

— It is mainly used in text classification that uses a high dimensional training dataset.

— Naive Bayes classifier is one of the simple & most effective classification algorithms which helps in building the fast machine learning models that can make quick predictions.

— It is a probabilistic classifier, which means it predicts on the basis of probability of an object.

— Some popular examples of Naive Bayes Algorithm are spam filtration, Sentimental analysis & classifying artides.

Assumptions of Naive Bayes

— Feature independence : The features of data are conditionally independent of each other, given the class label.

— Continuous label features : If a feature is continuous, then it is assumed to be normally distributed within each class.

— No missing data : The data should not contain any missing values.
— Features are equally important : All features are assumed to contribute equally to the prediction of class label.

Advantages of Naive Bayes
— easy to implement & computationally efficient.
— effective in cases with large no. of features.
— performs well even with limited training data.
— performs well in presence of categorical data.

Disadvantages of Naive Bayes.
— Assumes that features are independent, which may not always hold in real-world data.
— can be influenced by irrelevant attributes.
— may assign zero probability to unseen events, leading to poor generalization.

Applications.

— Spam Email filtering : classifies emails as spam/not spam.
— Text classification : Document & topic classification.
— Medical diagnosis : predicts the symptoms.
— Weather prediction : classifies weather on various factors.

Conclusion : Naive Bayes classifier proves effective in various applications; their efficiency & speed & ability to work with limited data makes them valuable in real-world scenarios.

```
.model_selection
.naive_bayes imp
.metrics import
plotlib.pyplot as
born as sns
das as pd
py as np
rnings

.filterwarnings('
the dataset
pd.read_csv("C:\

play dataset infor
(glass.head())
(glass.tail())
("Shape of datase
("Null values per

cting the distri
figure(figsize=(8
countplot(x='Type
title("Distribut
xlabel("Type")
ylabel("Count")
.show()  # Ensur

itialize the G
= GaussianNB()

split data into
= glass.drop(co
= glass['Type'

split the data
train, x_test

train the mo
.fit(x_train

Predict on
pred = nb.p
print("Predi

Calculate
accuracy =
print("Accu
```

Experiment no. 09

Aim : K-Nearest Neighbor Algorithm implementation.

Theory :

K-Nearest Neighbor (KNN) Algorithm.

- It is the learning where the value or result that we want to predict is within the training (labelled) data and the value which is in data we want to study is known as Target or Dependent Variable or Response Variable.
- K-Nearest Neighbor basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict to the nearest of the boundary line.
- Using the KNN algorithm, we fit the historical data and predict the future.

Why do we need KNN?

- It is a versatile and widely used machine learning algorithms that is used for its simplicity and ease of implementation.
- KNN algorithm works by finding the k-nearest neighbors to a given data point based on a distance metric such as Euclidean distance. The class or value of data point is then determined by the majority vote or average of k-neighbors. This approach allows the algorithm to adapt different patterns & make predictions based

on the local strudures of the data.

Advantages of KNN

- Easy to implement : The complexity of KNN is n very high, thus easier to implement.
- Adapts easily : KNN algorithm stores all the da in memory storage & hence whenever a new exampl or data point is added then algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- Few parameters : only parameters which are requ in training of a KNN algorithm are value of 'k' & choice of distance metric which we choose from me

Disadvantages

- Does not scale : KNN is considered a lazy algorith as it takes a lot of computing power as well as computing data storage.
- Dimensionality : It implies that KNN faces hard time classifying data points properly when dimensio is too high.
- Overfitting : As algorithm is affected due to the dimensionality as it is prone to the problem of overfitting.

Conclusion: KNN is a simple & powerful algorithm tha can be applied to both classification & regression problems. It is simple to understand, making no assumptions about data distribution.

Experiment no. 10

Aim : SVM algorithm implementation.

Theory :

SVM (support vector machines)

- SVM is a machine learning algorithm used for both classification & regression.
- The objective of SVM algorithm is to find a hyperplane in a N-dimensional space that distinctly classifies the data points.
- The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line.
- If the no. of input features is three, then the hyperplane becomes a 2-D plane.

Types of SVM.

SVM can be of two types —

(i) Linear SVM : Linear SVM is used for linearly seperable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly seperable data, and classifier is used called as linear SVM classifier.

(ii) Non-linear SVM : Non-linear SVM is used for non-linearly seperated data, which means if a

a dataset cannot be classified by using a straight line, then such data is termed as non-linear data & classifier used is called as Non-linear SVM classifier.

Hyperplane

There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

Support vectors

The data points or the vectors that are the closest to the hyperplane & which affects the position of the hyperplane are termed as the Support vector. since, these vectors support the hyperplane, hence is called a support vector.

Conclusion :-

Support vector machines (SVM) are powerful learning algorithms in machine learning, ideal for both classification and regression tasks. They excel at finding the optimal hyperplane for seperating data, making them suitable for applications like image classification and anomaly detection.

Thus, SVM is crucial for data scientists, as it enhances predictive accuracy & decision making across domains like artificial intelligence.