PROJECT

ABSTRACT

Handwritten digit prediction and classification are fundamental tasks in the field of machine learning and computer vision. This study presents an analysis of various machine learning models applied to the MNIST dataset, a widely recognized benchmark for handwritten digit recognition The objective is to evaluate the performance and compare different algorithms, including Convolution Neural Network(CNNs),Support Vector (SVMs),and Random Forests, in accurately predicting and classifying handwritten Digits. We explore preprocessing techniques, model architectures, hyperparameters, and performance metrics to understand the strengths and weaknesses of each approach. The results provide insights into the effectiveness and efficiency of these models for accurate handwritten digit classification, aiding in selecting methodologies for similar image classification tasks.

This research rigorously investigates the domain of handwritten digit prediction and classification, focusing on an extensive array of machine learning models applied to the MNIST dataset a pivotal benchmark for digit recognition. The study aims to comprehensively evaluate and compare the performance of diverse algorithms, notably Convolution Neural Network(CNNs), Support Vector Machines(SVMs),and Random Forests in accurately discerning digits from handwritten images. The analysis encompasses and in depth exploration of preprocessing techniques, model architectures, hyperparameters and a broad spectrum of performance metrics to delineate the strengths and weaknesses of each approach. The outcomes of this study provide profounds insights in to the efficiency and effectives of these models for precise handwritten digit classification, enabling informed decision making in selecting optimal methodologies for similar images classification task. This research stands as a comprehensive and authoritative reference for practitioners and researchers in the expansive realm of machine learning and computer vision.

I.1 Introduction

As a full time Beach Computer Engineering student at Karuna Institute of Technology and Sciences, I was provided an opportunity to undertake an internship at YBI Foundation ,New Delhi. Through this opportunity I got a very good experience on Machine Learning with pythonfor Business and Data Analytics both in terms of theoretical knowledge and some hands-on practical experiences with the help of building some projects with the theoretical knowledge that I gained on real world problems. While I am doing my 4th Semester, I was having Data Science subject which I was feeling hard to learn and those contents were like nearly dry concepts and I am not sure on why I was learning those things but through this internship I wasable relate those things that I studied theoretically in the class and with the things that I learnedin the internship while completing my assigned tasks.

1.2 Company Profile:

This training cum internship is offered by YBI Foundation which is a Delhi-based not-for- profit edutech company that aims to enable the youth to grow in the world of emerging technologies. They offer a mix of online and offline approaches to bring new skills, education, technologies for students, academicians and practitioners. They believe in the learning anywhere an in their academics and professions. d anytime approach to reach out to learners. The platform provides free online instructor-led classes for students to excel in data science, business analytics, machine learning, cloud computing and big data. They aim to focus on innovation, creativity, technology approach and keep themselves in sync with the present industry requirements. They endeavor to support learners to achieve the highest possible goalsin their academics and professions

1.3 Overview of Internship:

Python Programming with an emphasis on data structures, algorithms, and hand-written digit prediction-classification analysis involves a comprehensive exploration of key concept and practical applications. Here is an overview of how this internship entailed me in Business, Data Analytics and how to use Python to:

a. Introduction to Hand-written Digit Classification:

It will understand the problem of classifying Hand-Written digits and its
applications about the dataset, which often includes labeled images of hand-written digits.

b. Data Preprocessing and Feature Extraction:

The data preprocessing will load the data and preprocess the dataset, It will explore the data, handle missing values and perform basic statistics. The feature extraction will help covert completed data into a format that is easier for algorithm work with. **c. Visualize and communicate insights:**

Data visualization is an essential component of any data analysis project. Python provides several visualization libraries such as Matplotlib and Numpy that enable you to create informative and visually appealing charts and graphs.

d. Deploy learning models in Python Programming with Data Structures and Algorithms:

Once you have built a algorithms, you need to deploy it in a production environment. Python provides several tools such as Flask and Django for building web applications that can integrate machine learning models.

Overall, learning Python Programming with Data Structures and Algorithms internship can provide you with a competitive advantage in the job market and knowing how to apply it using Pythoncan help you stand out as a valuable candidate for data-driven roles.

1.4 Objective of the Report

The objective of Python Programming with Data Structures and Algorithms internship report is to showcase the knowledge and skills gained during the internship period. This report provides a comprehensive overview of the Python programming techniques that are used in the Data Structures and algorithms addressed during the internship.

Some specific objectives are:

1) During the internship, the primary focus was on addressing the data structure and objectives of project. The goals were established to achieve specific outcomes and milestones within the given timeframe.

Python Programming and Data Structures:

2) Python programming was applied extensively, leveraging data structures, learning algorithms, and techniques with the utilization of libraries such as Pandas, NumPy, and Scikit-learn.

Data Preparation and Feature Engineering:

3) The project involved a comprehensive process of data cleaning, pre-processing, and the application of feature engineering techniques. This was essential to prepare and structure the data for analysis.

Model Development and Evaluation:

4) Various Python programming models were developed, and their performance was assessed using appropriate evaluation metrics. Each model was scrutinized to understand its strengths and limitations.

Data Visualization and Dashboards:

5) Visualizations and interactive dashboards were created to effectively communicate the insights and recommendations derived from the data analysis. These graphical representations enhanced the understanding of the project's outcomes.

Key Learning Highlights:

6) The internship provided valuable insights into technical skill acquisition, insights into business operations, and the practical application of machine learning techniques to real-world problems.

Recommendations for Enhancements:

7) Based on the obtained results, recommendations were formulated for potential improvements and enhancements to the project or models, aiming to optimize outcomes further.

II. Background

2.1 What is Python Programming:

Python supports multiple programming paradigms, including object-oriented, imperative, functional,and procedural programming. It has gained immense popularity due to its clear and concise syntax, extensivestandard library, and a vast ecosystem of third-party libraries and frameworks.

Key features of Python include:

1. **Readability**:

Python emphasizes code readability and uses a clean and consistent syntax, making it easier to write and understand programs.

2. **Interpretation**:

Python is an interpreted language, which means that you don't need to compile your code before running it. The Python interpreter translates the code into machine-readable bytecode on thefly.

3. **Dynamic Typing**:

Python is dynamically typed, meaning you don't need to declare the data type of a variable explicitly. The interpreter infers the type based on the context.

4. **Indentation**:

   Python uses indentation (whitespace) to define blocks of code. This enforces a consistent coding style and eliminates the need for explicit block delimiters e.g., braces.

5. **Rich Standard Library:**

   Python comes with a large standard library that provides a wide range of modules and functions for tasks such as file handling, networking, web development, data manipulation, and more.

6. **Portability**:

   Python is available for various platforms, including Windows, macOS, Linux, and more. Programs written in Python can run on different operating systems without modification.

7. **Object-Oriented:**

   Python supports object-oriented programming, allowing you to define and work with classesand objects for code organization and reuse.

Python is widely used in various domains, including web development, data analysis, artificial intelligence,machine learning, scientific computing, automation, and more. It is a go-to language for many programmersdue to its versatility, ease of learning, and applicability in a wide range of projects.

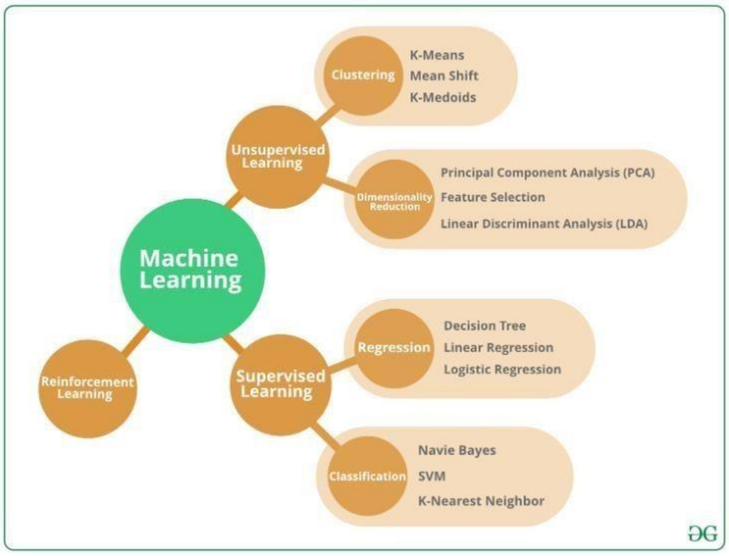
2.2 What is Machine Learning

Machine learning is a subset of artificial intelligence that involves developing algorithms and statistical models that allow computer systems to automatically learn from data without being explicitly programmed.

In traditional computer programming, a set of rules and instructions are written by programmers to enable the computer to perform a specific task. However, in machine learning, algorithms are designed to identify patterns and relationships in data and use this information to make predictions or decisions about new data.

The machine learning process involves several steps, including data collection and pre-processing, feature selection and engineering, model selection and training, and model evaluation and optimization. The models can be supervised, unsupervised, or semi-supervised, depending on the availability of labelled data during training.

Machine learning has many applications, including natural language processing, image recognition, fraud detection, predictive maintenance, and recommendation systems, among others.With the availability of large amounts of data and the advancements in computing power and machine learning algorithms, machine learning is becoming increasingly popular and powerful, allowing for more accurate predictions and insights from data.

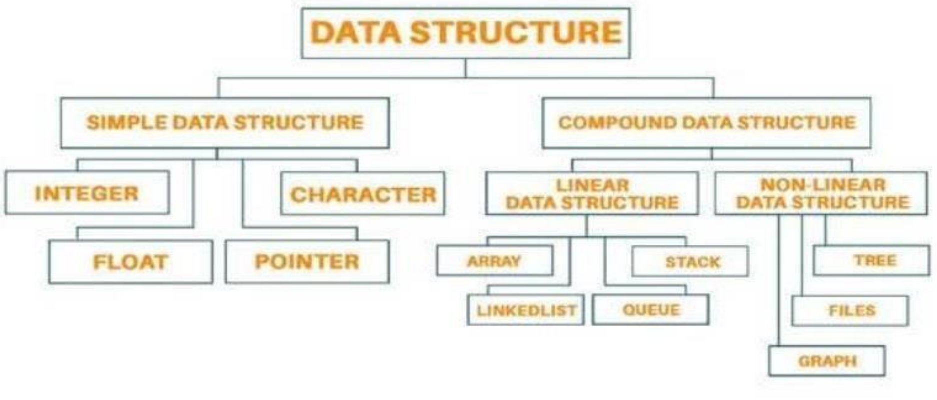


2.3 How it used in Data Structures:

Data structures are fundamental constructs in computer science that organize, store, and manipulate data in a way that enables efficient access and modification. They serve as the building blocks for designing algorithms and solving complex computational problems. Data structures can be classified into several types, each with unique characteristics and use cases.

Arrays, one of the simplest data structures, store elements in contiguous memory locations, allowing for quick access through indices. Linked lists, on the other hand, consist of nodes where each node contains both data and a reference to the next node, enabling dynamic andefficient insertion and deletion operations. Stacks and queues are abstract data types that restrict how elements are added or removed, following Last-In-First-Out (LIFO) and First-In-First-Out (FIFO) principles, respectively. Trees and graphs organize data hierarchically, representing relationships between elements, making them useful for tasks like hierarchical data representationand network modeling. Hash tables employ hash functions to map keys to specific locations, facilitating rapid data retrieval based on those keys. Choosing the appropriate data structure depends on the nature of the data, the operations to be performed, and the efficiency requirementsof the algorithm or application at hand

1) **Arrays:** A collection of elements stored in contiguous memory locations, each identified by anindex or a key
2) **Linked List**: A collection of nodes, where each node contains data and a refer(or link) to the nextnode in the sequence
3) **Stacks**: A last in, First out data structure, where elements are added and removed from the sameend called the top.
4) **Queues**: A first in ,First Out data structure ,Where elements are added at the rear and removed fromthe front



2.4 How Data Structures and Algorithm used in Machine Learning?

Data structures play a crucial role in machine learning, enabling efficient data storage, manipulation, and processing. Here are some key ways data structures are used in machine learning:

1. **Data Representation:**

   In machine learning, data is typically represented using arrays, matrices, or tensors. These data structures allow for the efficient storage of features and labels. For example, in image recognition, images are often represented as multi-dimensional arrays where each element corresponds to a pixel value.

2. **Data Preprocessing:**

Before feeding data into machine learning algorithms, preprocessing steps such as normalization, scaling, and feature extraction are applied. Data structures like arrays are used to store intermediate results during preprocessing. Techniques like Principal Component Analysis (PCA) use matrices for dimensionality reduction.

3. **Data Storage**:

   Databases and data structures like hash tables are used to store and retrieve large datasets. Efficient data storage is crucial for handling big data, which is common in machine learning applications.

4. **Graphs for Modeling Relationships:**

   Graph data structures are employed in various machine learning applications, such as social network analysis and recommendation systems. Nodes and edges represent entities and relationships between them, allowing algorithms to understand intricate patterns and connections in the data.

5. **Trees for Decision Making**:

   Decision trees, a type of tree data structure, are widely used in machine learning for classification and regression tasks. They represent decisions and their possible consequences in a tree-like model, allowing for efficient decision-making processes.

6. **Sparse Data Structures:**

   Machine learning datasets often contain a lot of empty or zero values. Sparse data structures, like Compressed Sparse Row (CSR) matrices, are employed to efficiently store and process these datasets, saving memory and computational resources.

7. **Queues for Batch Processing:**

   In training machine learning models, data is often processed in batches. Queues are used to manage and process these batches in an orderly manner, ensuring efficient training and minimizing memory usage.

8. **Efficient Search and Retrieval:**

   Data structures like kd-trees and ball trees are used in machine learning algorithms, such as k-nearest neighbors (KNN), to efficiently search and retrieve similar data points, making them essential for tasks like clustering and recommendation systems

III. Technologies Used in Project

3.1 Programming Language:

Programming Language used in the project is Python Programming. Python provides a wide range of libraries for machine learning, including scikit-learn, TensorFlow, Keras, PyTorch, and others. These libraries provide efficient implementations of popular machine learning algorithms, such as linear regression, decision trees, random forests, support vector machines, and neural networks, among others.

3.2 Scikit-learn Library:

Scikit-learn provides a wide range of machine learning algorithms, including regression, classification, clustering, and dimensionality reduction. It also provides tools for preprocessing data, model selection, and evaluation. Some of the popular algorithms available in scikit-learn include linear regression, logistic regression, support vector machines, decision trees, random forests, K-nearest neighbors, and neural networks, among others.

3.3 NumPy and Pandas Libraries:

NumPy provides support for arrays and matrices, which are essential data structures for scientific computing. It provides fast and efficient implementationsof mathematical operations on arrays and matrices, making it well-suited for numericalcomputing. NumPy also provides tools for linear algebra, Fourier transforms, and random number generation, among others. Pandas, on the other hand, is a library for data manipulationand analysis. It provides support for working with tabular data, such as spreadsheets or databases, and provides tools for data cleaning, aggregation, and merging. Pandas also providessupport for time-series data, making it well-suited for analyzing financial data or other time- based data.

Jupyter Notebook:

Jupyter Notebook is an open-source web-based application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports a wide range of programming languages, including Python, R, and Julia, among others.

Classification Algorithms:

Classification algorithms are a fundamental aspect of supervised machine learning, where the goal is to assign predefined categories or labels to input data points based on their features. One of the simplest yet widely used algorithms is **Logistic Regression**, a linear classification method that estimates the probability that an input point belongs to a particular class. It models the relationship between the features and the binary outcome using the logistic function, which maps any real-valued number into the range of 0 and 1. Logistic Regression is especially useful for binary classification tasks, such as spam detection in emails or customer churn prediction in business, where it predicts whether an event will occur (class 1) or not (class 0) based on the input features. It's computationally efficient, interpretable, and serves as a foundation for more complex algorithms.

Moving to more complex models, **Decision Trees** provide a non-linear approach to classification by recursively splitting the data based on feature thresholds, creating a tree-like structure where each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents the predicted class label. Decision trees are intuitive and easy to visualize, making them valuable for understanding the decision-making process. An extension of this, **Random Forest**, leverages the power of ensemble learning by combining multiple decision trees. Each tree is trained on a random subset of the data, and the final prediction is determined by the majority vote of all individual trees. This ensemble technique enhances accuracy and generalization, making Random Forest a robust choice for classification tasks, especially when dealing with noisy or high-dimensional data. Random Forest mitigates overfitting, a common challenge in machine learning, and is highly effective in applications like image recognition and fraud detection where complex patterns need to be identified.

Random Forest Model:

A Random Forest is an ensemble learning method used for both classification and regression tasks in machine learning. It operates by constructing a multitude of decision trees during training and outputs the class (for classification) or the mean prediction (for regression) of the individual trees.

Key Characteristics:

1. **Ensemble Learning:**

Random Forest is an ensemble method because it combines multiple individual models (decision trees in this case) to create a more accurate and robust predictive model. Each tree is trained independently on a different subset of the data, and their predictions are combined to make the final prediction.

2. **Decision Trees:**

The base models in a Random Forest are decision trees. Decision trees make decisions based on asking a series of questions and use the answers to classify the data points or predict outcomes. In Random Forest, each tree is trained on a random subset of the data, and the randomness reduces the risk of overfitting.

3. **Bootstrapping:**

Random Forest uses a technique called bootstrapping, where subsets of the training data are created with replacement. This means that each tree in the forest is trained on a slightly different dataset, adding diversity to the ensemble.

4. **Feature Randomness:**

In addition to using different subsets of data, Random Forest introduces randomness at the feature level. When considering a split in a decision tree, the algorithm does not consider all features but only a random subset of features. This prevents certain features from dominating the decision-making process.

5. **Voting/Averaging:**

For classification tasks, Random Forest uses majority voting, where each tree "votes" for a class, and the class with the most votes become the predicted class. For regression tasks, the predictions of individual trees are averaged to obtain the final prediction.

IV. Implementation of Internship Project

4.1 Problem Statement:

In the context of machine learning, the problem of Handwritten Digit Prediction, also known as the MNIST digit classification problem, is a classic and well-studied task. The goal is to develop an algorithm that can accurately classify images of handwritten digits (0 through 9) into their respective numerical labels. The dataset used for this task, called the MNIST dataset, consists of thousands of 28x28 pixel grayscale images of handwritten digits, making it a widely-used benchmark in the machine learning community.

4.2. About the Dataset:

The MNIST dataset is a large collection of handwritten digits widely used for training and testing machine learning models in the field of computer vision and pattern recognition. MNIST stands for Modified National Institute of Standards and Technology, indicating its origin from the United States National Institute of Standards and Technology (NIST). The dataset was modified and curated to suit the needs of the machine learning community.

The MNIST dataset comprises 60,000 training images and 10,000 testing images, each containing a grayscale handwritten digit (0 through 9). These images are 28x28 pixels in size, resulting in a total of 784 features (pixels) per image. Each image is labeled with the corresponding digit it represents, making it a labeled dataset. The dataset is balanced, meaning there is an equal distribution of digits across the training and testing sets, ensuring that the model learns to recognize all digits equally well.

4.3. Procedure:

1. **Load the dataset**:

   Load the Galton dataset using the Pandas library.

2. **Pre-process the data:**

   Pre-process the data by removing any missing values or outliers.

3. **Split the data**:

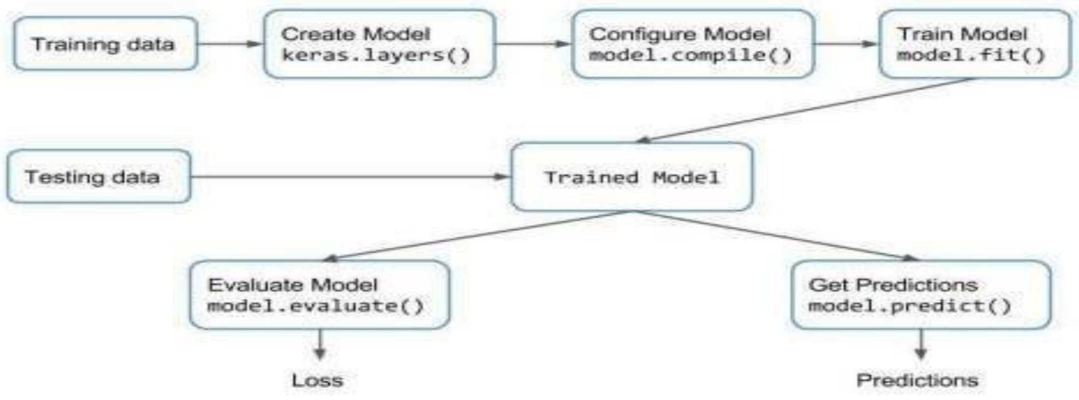
   Split the data into training and testing sets. The training set is used to trainthe model, and the testing set is used to evaluate its performance.

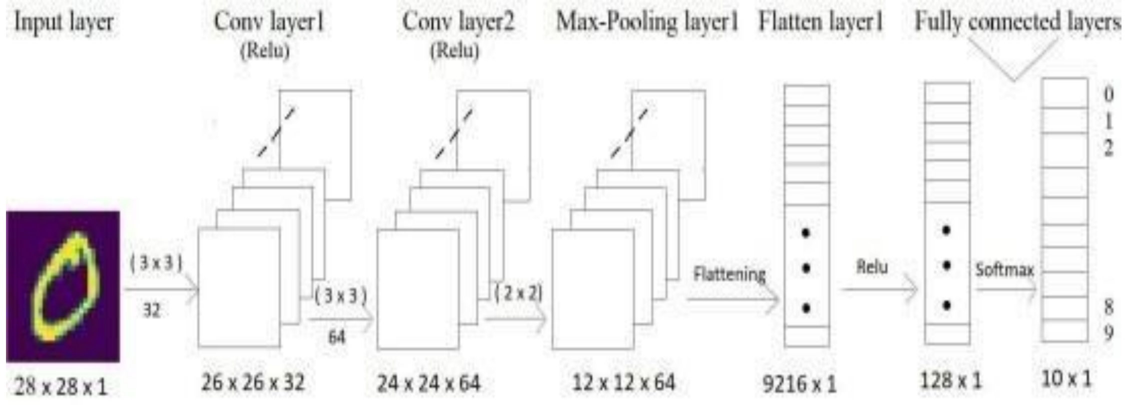
4. **Feature engineering**:

   Feature engineering is the process of selecting and transforming thefeatures that will be used in the model. In this case, we can select the height of the father andmother as the input features and the height of the child as the output feature.

5. **Train the model**:

   Train a classification model using the Scikit-learn library. We can use different algorithms such as logistic regression, decision tree classification, or random forest classification to train the model.


Training data
Create Model
keras.layers()
Configure Model
model.compile()
Train Model
model.fit()
Testing data
Trained Model
Evaluate Model
model.evaluate()
Get Predictions
model.predict()
Loss
Predictions


Input layer
Conv layer1
(Relu)
Conv layer2
(Relu)
Max-Pooling layer1
Flatten layer1
Fully connected layers
(3 x 3)
32
(3 x 3)
64
(2 x 2)
Flattening
Relu
Softmax
0
1
2
8
9
28 x 28 x 1
26 x 26 x 32
24 x 24 x 64
12 x 12 x 64
9216 x 1
128 x 1
10 x 1

4.4. Code:

```
import pandas as pd import numpy
as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits

#Importing the dataset df=load_digits()

_,    axes=plt.subplots(nrows=1,ncols=4,figsize=(10,3))    for
ax,image,label in zip(axes,df.images,df.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title("Training: %i" % label)
n_samples=len(df.images) data=
df.images.reshape((n_samples,-1))

#Scaling the data for data preprocessing data=data/16

#Splitting the data for training and testing
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(data,df.target,test_size=0.3)
X_train.shape, X_test.shape,y_train.shape,y_test.shape

#Building the Classification Model
from   sklearn.ensemble   import   RandomForestClassifier
rf=RandomForestClassifier()       rf.fit(X_train,y_train)
y_pred =rf.predict(X_test) y_pred

#Finding the Model Accuracy
from sklearn.metrics import confusion_matrix,classification_report
confusion_matrix(y_test,y_pred)
print(classification_report(y_test,y_pred))
```

V. Outcomes/ Result

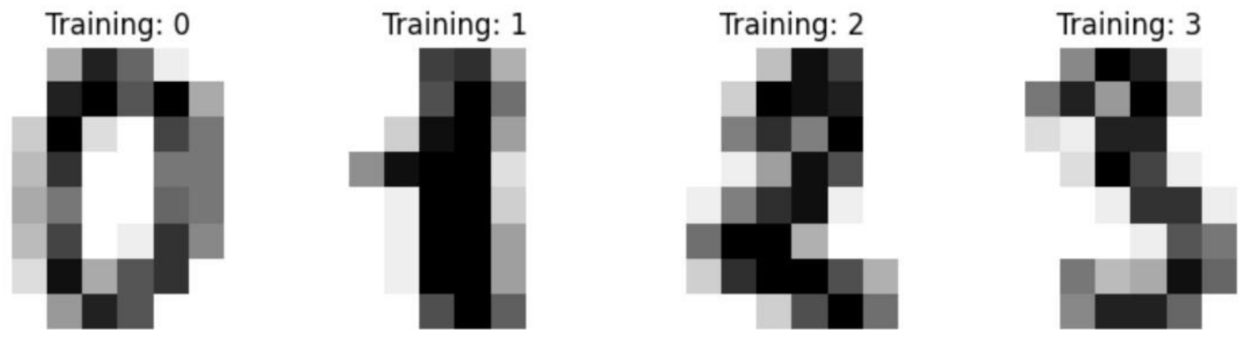
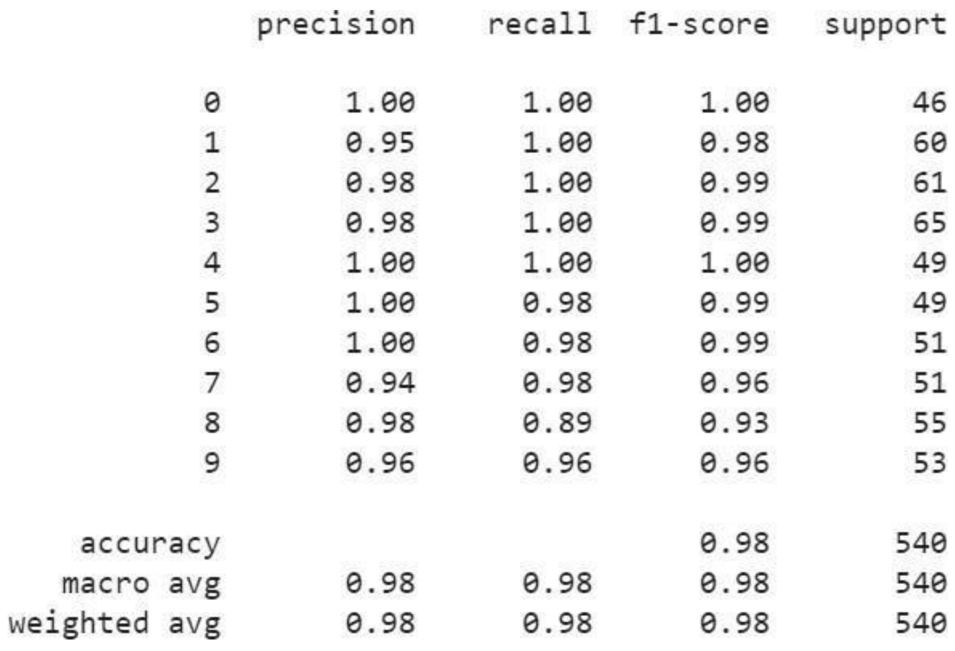


Fig.1.1

In Fig.1.1, Each image corresponds to a handwritten digit (0 through 9) and is displayed in a separate subplot. The titles of the subplots indicate the training labels of the displayed digits. This visualization is helpful for understanding the structure of the dataset and verifying that the loaded images match their corresponding labels.



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        46
           1       0.95      1.00      0.98        60
           2       0.98      1.00      0.99        61
           3       0.98      1.00      0.99        65
           4       1.00      1.00      1.00        49
           5       1.00      0.98      0.99        49
           6       1.00      0.98      0.99        51
           7       0.94      0.98      0.96        51
           8       0.98      0.89      0.93        55
           9       0.96      0.96      0.96        53

    accuracy                           0.98       540
   macro avg       0.98      0.98      0.98       540
weighted avg       0.98      0.98      0.98       540
```

Fig.1.2

Fig.1.2 displays the classification report of the random forest model. A classification report is a performance evaluation metric in machine learning that provides a summary of various metrics to assess the quality of a classification model. It is particularly useful for problems where the classes are imbalanced. The classification report is typically generated after a model has been trained and tested on a dataset and is a crucial tool for evaluating the model's performance, especially in tasks like binary and multiclass classification. The classification report includes the following metrics for each class in the dataset:

Precision: Precision measures the accuracy of positive predictions. It is calculated as the ratio of true positive predictions to the total number of positive predictions (true positives + falsepositives). A high precision indicates that the class predictions are accurate and reliable.

Recall (Sensitivity or True Positive Rate): Recall measures the ability of the model to correctly identify all relevant instances in a class. It is calculated as the ratio of true positive predictions to the total number of actual positives (true positives + false negatives). High recall means the model is good at capturing all instances of the class.

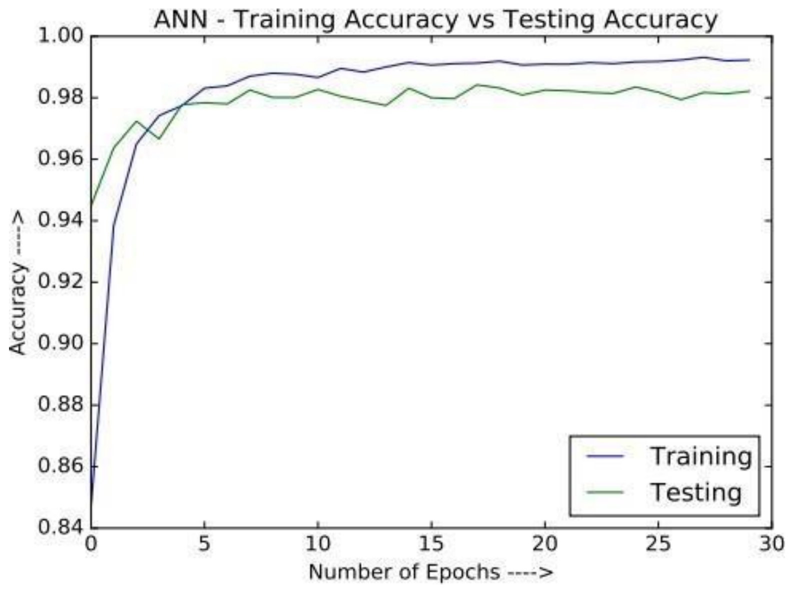
F1-Score: F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, giving equal weight to both metrics. F1-score is especially useful when the class distribution is imbalanced.

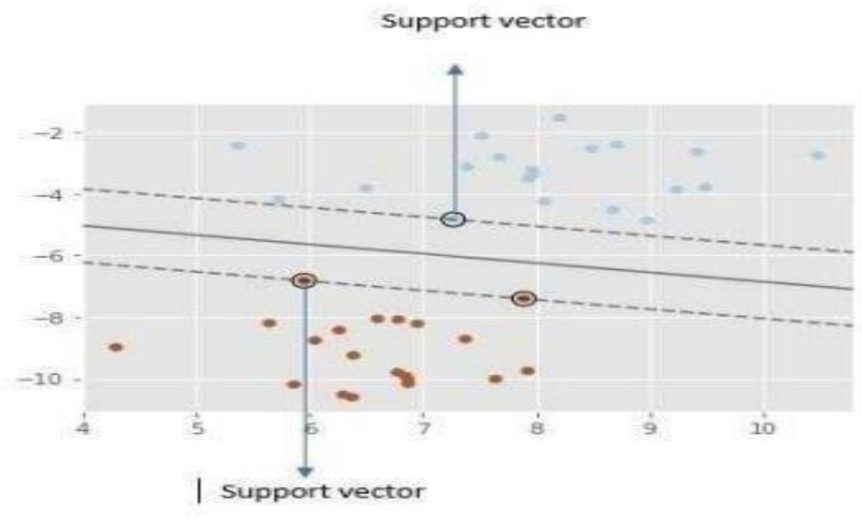
Support: Support is the number of actual occurrences of the class in the specified dataset. It helps in understanding how many actual instances of the class are present.

Accuracy: While not specific to a class, accuracy measures the overall correctness of the model and is calculated as the ratio of correct predictions to the total number of predictions. It's a useful metric for balanced class distributions but can be misleading in imbalanced datasets.

Here the values of precision, Recall, F1-Score for 9 classes are nearly 1 and there is a good support value. The accuracy of the model is 98% and the model is very much well performing on the given dataset and the model prediction were good.

Algorithm	**Accuracy**
Logistic Regression	0.98
Decision Tree Classifier	0.99(nearly 1.0)
Support Vector Machine	0.98888





VI. Conclusion

The report concludes the implementation of an internship project related to the Galton height problem, which is a statistical problem Hand written.Digit Prediction and Classification Analysis provide a foundational understanding of machine learning and computer vision techniques and serve as a stepping stone for more complex image recognition and classification tasks. The continuous refinement of models and algorithms ensures increasingly accurate and efficient predictions, with applications spanning from digit recognition to broader real-world problems.as pre-processing the data, feature engineering, and training a regression model using different algorithms such as logistic regression, Rain Forcement and support Vector machine. The performance of the model is evaluated using metrics such as Accuracy score, Precision

Score, and Recall score. Overall, the project demonstrates the application of statistical concepts and techniques to a real-world problem, and highlights the importance of data pre-processing and feature engineering in building accurate predictive models. Hence the decision tree classification model is performing well than other two models.

VII. References

[1].Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[2]. Ishwaran, H., & Kogalur, U. B. (2007). Random survival forests. The Annals of Statistics, 35(6), 2557-2584.

[3]. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R News, 2(3), 18-22.

[4]. Probst, P., Wright, M. N., & Boulesteix, A. L. (2019). Hyperparameters and tuning strategies for random forest. Wiley StatsRef: Statistics Reference Online.

[5]. Belgiu, M., Drăguţ, L., & Strobl, J. (2014). Random forests in remote sensing: A review of applications and future directions. ISPRS Journal of Photogrammetry and Remote Sensing, 94, 24-31.

[6]. Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Lawrence, J., ... & Tibshirani, R. (2012). Random forests for classification in ecology. Ecology, 93(11), 2783-2792.

[7].Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., & Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. ISPRS Journal of Photogrammetry and Remote Sensing, 67(1), 93-104.

[8]. Belgiu, M., & Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and perspectives. Forests, 7(6), 122.

[9].Breiman, L. (2003). Statistical modeling: The two cultures (with comments and a rejoinder by the author). Statistical Science, 18(4), 199-231.

[10]. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R News, 2(3), 18-22.

[11]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[12]. R Core Team. (2023). R: A language and environment for statistical computing. R Foundation for Statistical Computing. https://www.R-project.org/