Handwritten Digit Recognition
Project Report

Rucchik Dilip Rangari
24b2248

This report presents a comprehensive and structured overview of my learning journey during the Winter in Data Science (WiDS) program. Spanning a period of four weeks, the program offered systematic and progressive exposure to core concepts in data science, beginning with Python programming and essential data analysis libraries, and advancing toward machine learning and deep learning methodologies. The curriculum covered fundamental machine learning principles, supervised learning techniques such as regression and classification, unsupervised learning methods, artificial neural networks, and convolutional neural networks, culminating in a final hands-on project that integrated both theoretical and practical knowledge.

Throughout the program, emphasis was placed on developing a strong conceptual foundation alongside practical implementation through coding experiments, data preprocessing, model training, evaluation, and performance analysis. This report documents the theoretical concepts studied, experimental implementations performed during learning and execution, and the insights gained from iterative experimentation and problem-solving. The primary objective of this report is to demonstrate conceptual clarity, analytical thinking, practical proficiency, and an academic understanding of modern data science workflows, while reflecting on the learning outcomes achieved through the WiDS program.

1. INTRODUCTION

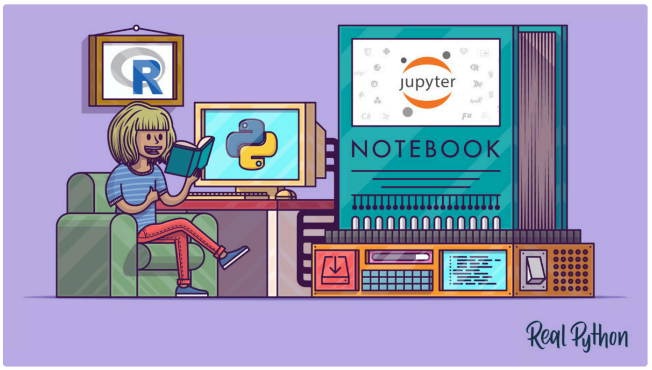
Data Science has emerged as a significant interdisciplinary field that combines statistics, computer science, and domain knowledge to extract meaningful insights from data. With the increasing availability of large and complex datasets, data-driven approaches have become essential for analysis, prediction, and decision-making across various domains. As a result, data science and machine learning skills are now widely applied in both academic research and industry.

The Winter in Data Science (WiDS) program was designed to introduce learners to the core concepts and practical applications of data science in a structured and progressive manner. This report serves as a formal academic documentation of my learning experiences throughout the WiDS program, including theoretical understanding, experimental implementation, observations, and challenges encountered during the learning process.

The program was conducted over four weeks, with each week focusing on increasingly advanced topics. The curriculum began with Python programming and essential data science libraries, followed by supervised learning techniques such as regression and classification. It then progressed to artificial neural networks and unsupervised learning methods, and finally to deep learning using convolutional neural networks (CNNs). Hands-on coding exercises, experiments, and a final project reinforced the theoretical concepts and provided practical exposure to end-to-end data science workflows.

2. WEEK 1: PYTHON LIBRARIES AND INTRODUCTION TO MACHINE LEARNING

2.1 Jupyter Notebook



Jupyter Notebook: An Introduction

Jupyter Notebook is used by launching it from Anaconda Prompt or terminal.

Jupyter Notebook was introduced as the primary development environment for implementing and experimenting with Python code. It provides an interactive platform that allows code execution in small,

manageable cells along with immediate visualization of outputs. The ability to combine code, text, and graphical results in a single interface made Jupyter Notebook particularly useful for exploratory data analysis, debugging, and documenting experiments.

2.1 Python Programming Foundations

Python code is written directly inside Jupyter Notebook cells. Python is a versatile and widely adopted programming language in the field of data science due to its simplicity, readability, and extensive library support. During Week 1, emphasis was placed on strengthening fundamental Python concepts such as variables, data types, conditional statements, loops, functions, and basic data structures. These concepts formed the foundation for writing efficient and modular code, which is essential for implementing data analysis and machine learning algorithms in later stages of the program.

2.3 NumPy

NumPy is a core Python library used for numerical and scientific computing. During this week, NumPy was used to work with multi-dimensional arrays and perform efficient mathematical operations. Concepts such as array creation, indexing, slicing, broadcasting, and vectorized computations were explored. The use of NumPy significantly improved computational performance compared to traditional Python loops, especially when working with large datasets.
NumPy is imported using the `import` statement and is commonly aliased as `np`

2.4 Pandas

Pandas was introduced as a powerful library for data manipulation and analysis. The use of Series and DataFrames enabled structured handling of tabular data. Various operations such as data loading, filtering, sorting, grouping, aggregation, and handling missing values were performed. Pandas played a crucial role in data cleaning and preprocessing, which are essential steps before applying machine learning models.
Pandas is usually imported as `pd`. (`import` pandas `as` pd)

2.5 Matplotlib

Matplotlib was used for basic data visualization to represent data in graphical form. Line plots, bar charts, histograms, and scatter plots were created to understand data distributions and trends. Visualization using Matplotlib helped transform raw numerical data into interpretable visual insights, aiding in better data understanding.
Imported using (import matplotlib.pyplot as plt)

2.6 Seaborn

Seaborn is a statistical data visualization library built on top of Matplotlib and was used to create more advanced and visually appealing plots. It was applied to generate box plots, heatmaps, and distribution plots that helped identify correlations, patterns, and outliers in datasets. Seaborn enhanced the interpretability of data by providing clearer and more informative visual representations.
Imported using (import seaborn as sns)

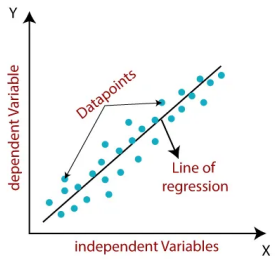
Week 2 – Supervised Learning (Regression + Classification)

3.1 Introduction to Supervised Learning

Week 2 focused on **supervised learning**, where models are trained using labeled datasets to predict outcomes. The primary goal was to understand how relationships between input features and output labels are learned and how these models can be evaluated. Emphasis was placed on regression techniques for continuous outputs and classification techniques for categorical outputs.

3.2 Regression

Regression is a supervised learning technique used to predict continuous numerical values. During this week, regression was introduced to model the relationship between dependent and independent variables. The objective was to fit a mathematical function that best represents the data.



How it is called and used (conceptually):

1. Input features (X) and output values (y) are defined
2. The model learns coefficients that minimize prediction error
3. Model performance is evaluated using error metrics

3.3 Multiple Linear Regression

Multiple Linear Regression is an extension of simple linear regression where multiple independent variables are used to predict a single dependent variable. This approach helps capture more complex relationships within the data.

How it is called and used:

1. Multiple features are provided as input
2. The model learns multiple coefficients (weights)
3. Predictions are made using a linear combination of inputs

General form:

$y=b0+b1x1+b2x2+\cdots+bnxn$

3.4 Scikit-learn (Sklearn)

Scikit-learn is a widely used Python library for machine learning that provides simple and efficient tools for data analysis and modeling. During Week 2, Scikit-learn was used to implement regression and classification models in a standardized workflow.

How Scikit-learn is called and used:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Typical workflow in Scikit-learn:

1. Import the required model
2. Split data into training and testing sets
3. Train the model using `.fit()`
4. Make predictions using `.predict()`
5. Evaluate model performance

3.5 Logistic Regression

Logistic Regression is a supervised classification algorithm used to predict categorical outcomes, particularly binary classes. Instead of predicting continuous values, it estimates the probability of class membership using a sigmoid function.

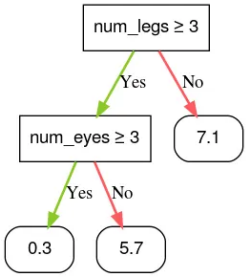
How it is called and used:

```
from sklearn.linear_model import LogisticRegression
```

1.Used for binary classification problems
2.Outputs probabilities between 0 and 1
3.Evaluated using accuracy, confusion matrix, precision, and recall

3.6 Decision Trees

Decision Trees are supervised learning models that use a tree-like structure to make decisions based on feature values. They work by recursively splitting data into subsets based on conditions that maximize information gain.



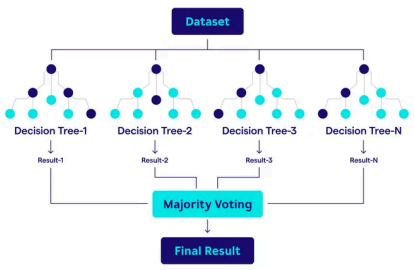
How it is called and used:

```
from sklearn.tree import DecisionTreeClassifier
```

1. Easy to interpret and visualize
2. Can handle both classification and regression tasks
3. Prone to overfitting if not controlled

3.7 Random Forests

Random Forest is an ensemble learning technique that combines multiple decision trees to improve prediction accuracy and reduce overfitting. Each tree is trained on a random subset of data and features, and the final output is determined by majority voting or averaging.



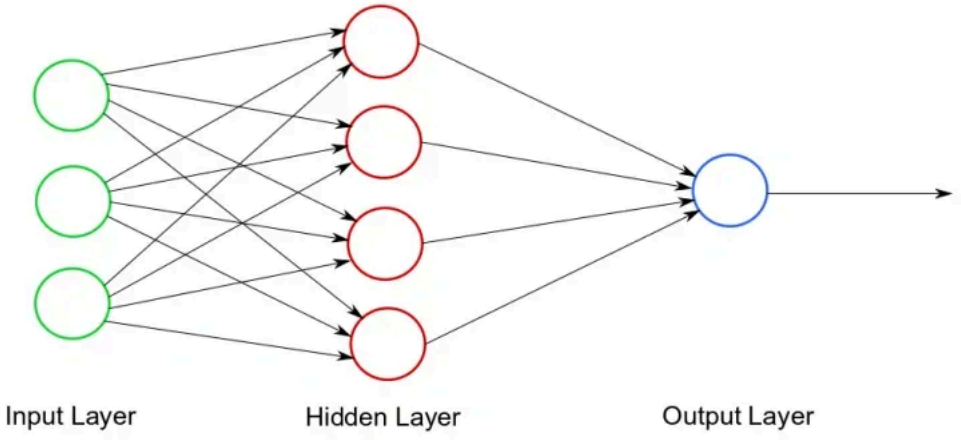
How it is called and used:

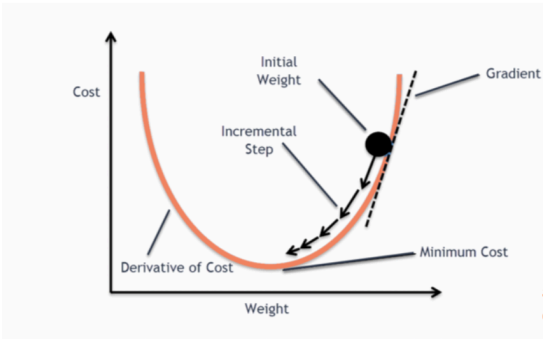
```
from sklearn.ensemble import RandomForestClassifier
```

1. More robust than a single decision tree
2. Handles large datasets efficiently
3. Provides better generalization performance

4. WEEK 3: NEURAL NETWORKS AND UNSUPERVISED LEARNING

4.1 Neural Networks



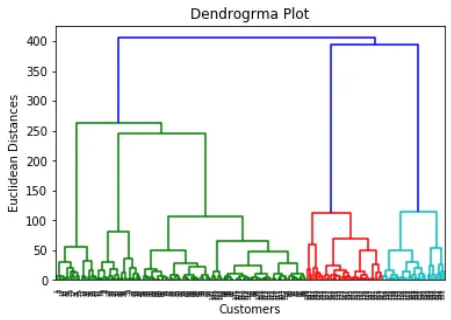


Neural Networks are computational models inspired by the structure and functioning of the human brain. They consist of interconnected layers of neurons that process input data through weighted connections and activation functions. During Week 3, the fundamental components of neural networks, including neurons, weights, bias, activation functions, and loss functions, were studied. The concepts of forward propagation and backpropagation were introduced to understand how neural networks learn by minimizing error during training. Neural networks were implemented to perform supervised learning tasks such as classification and regression.

4.2 Unsupervised Learning

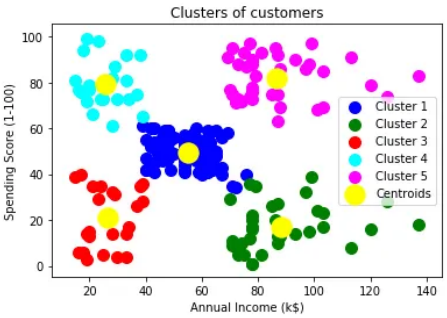
Unsupervised learning involves analyzing data without labeled outputs to discover hidden patterns or structures within the data. This week introduced the motivation and importance of unsupervised learning, particularly for exploratory data analysis. Unlike supervised learning, these techniques focus on grouping, clustering, or reducing the dimensionality of data based solely on inherent similarities.

4.3 Hierarchical Clustering



Hierarchical clustering is an unsupervised learning technique that builds a hierarchy of clusters either using a bottom-up (agglomerative) or top-down (divisive) approach. During this week, agglomerative hierarchical clustering was studied in detail. The method groups data points based on distance metrics and linkage criteria, and the results are commonly visualized using dendrograms. Hierarchical clustering is useful for understanding nested cluster relationships and does not require the number of clusters to be predefined.

4.4 K-Means Clustering



K-Means clustering is one of the most widely used unsupervised learning algorithms for partitioning data into a fixed number of clusters. The algorithm iteratively assigns data points to the nearest cluster centroid and updates centroids until convergence. During the learning process, emphasis was placed on selecting the appropriate number of clusters using methods such as the elbow method. K-Means was observed to be computationally efficient and effective for well-separated clusters.

4.6 Dimensionality Reduction

Dimensionality reduction techniques are used to reduce the number of input features while preserving important information. These methods help improve model performance, reduce computational complexity, and enhance data visualization.

4.6.1 Principal Component Analysis (PCA)

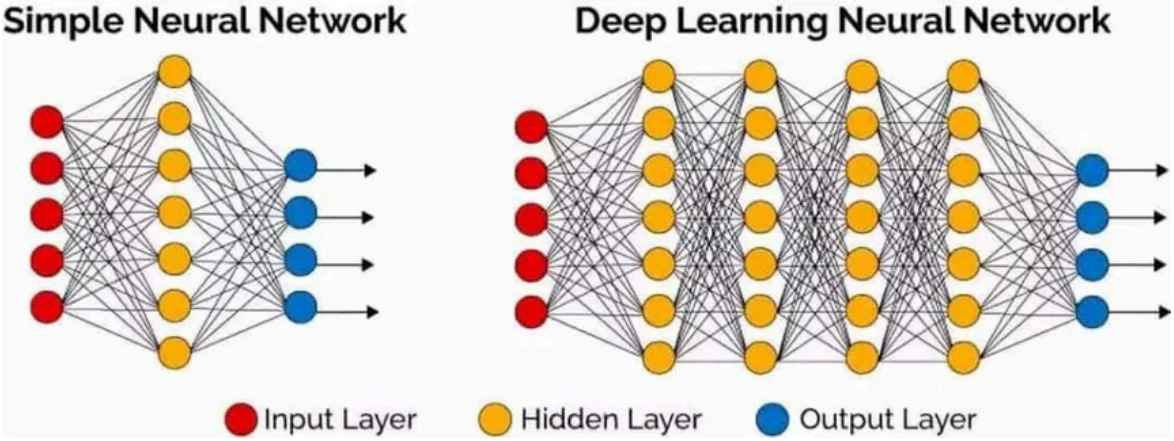
Principal Component Analysis is a statistical technique used to transform high-dimensional data into a lower-dimensional space by maximizing variance. PCA identifies orthogonal components that capture the most significant variations in the data. During this week, PCA was applied for feature reduction and data visualization, making complex datasets easier to interpret.

4.6.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is a supervised dimensionality reduction technique that aims to maximize class separability. Unlike PCA, which focuses on variance, LDA considers class labels to project data onto a lower-dimensional space. LDA was studied to understand how dimensionality reduction can improve classification performance by enhancing class discrimination.

5. WEEK 4: DEEP LEARNING AND CONVOLUTIONAL NEURAL NETWORKS

5.1 Introduction to Deep Learning



Deep Learning is a subset of machine learning that uses multi-layered neural networks to learn complex patterns from large datasets. Unlike traditional machine learning algorithms, deep learning models automatically learn hierarchical feature representations from raw data. During Week 4, deep learning concepts were studied with a focus on practical implementation using the TensorFlow framework. This week emphasized understanding model architecture, training processes, and performance evaluation for complex data types such as images.

5.2 TensorFlow Framework

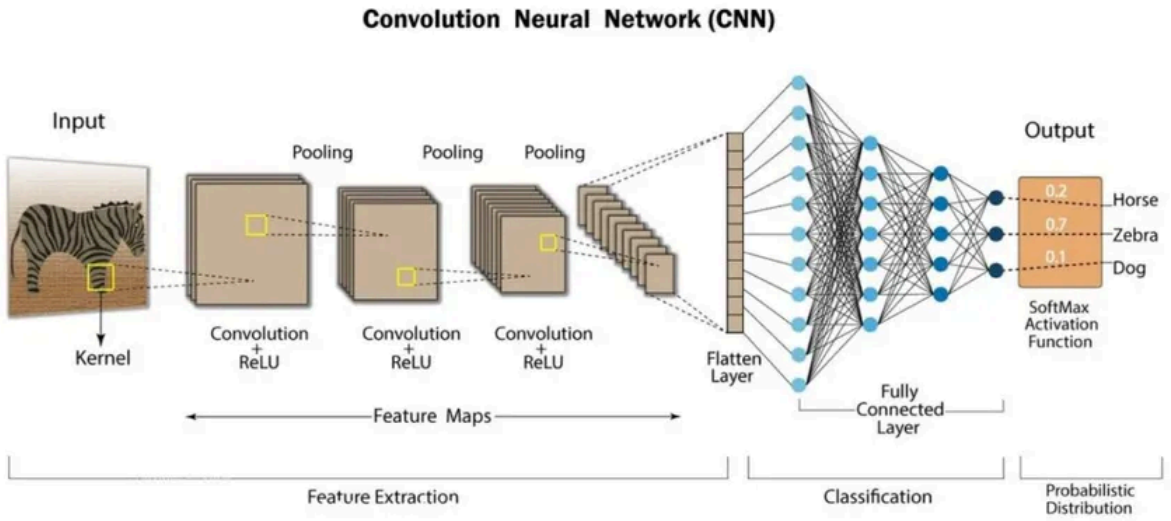
TensorFlow is an open-source deep learning framework widely used for building, training, and deploying neural network models. During this week, TensorFlow was used to implement deep learning models efficiently using high-level APIs such as Keras. The framework provided tools for defining model architectures, compiling models with appropriate loss functions and optimizers, and training models using labeled datasets. TensorFlow enabled faster experimentation and scalable model development.

5.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a specialized class of deep learning models designed primarily for image and visual data analysis. CNNs automatically extract spatial features from input images using convolutional layers. Key components such as convolution layers, filters, kernels, feature maps, pooling layers, and fully connected layers were studied. CNNs were observed to outperform traditional neural networks in image classification tasks due to their ability to capture local spatial patterns.

5.4 Working of CNN

The working of a CNN involves multiple stages of feature extraction and classification. Initially, convolution layers apply filters to the input image to detect low-level features such as edges and textures. Pooling layers reduce spatial dimensions and computational complexity while retaining important features. As the network deepens, higher-level features such as shapes and objects are learned. Finally, fully connected layers perform classification based on the extracted features.



6. CONCLUSION

The Women in Data Science (WiDS) program provided a strong and structured foundation in data science and machine learning through a well-designed, progressive curriculum. Beginning with Python programming and essential data science libraries, and advancing through supervised learning, unsupervised learning, neural networks, and deep learning, the program enabled a comprehensive understanding of both fundamental and advanced concepts. The step-by-step learning approach ensured conceptual clarity while gradually introducing increasing levels of complexity.

The integration of hands-on coding exercises, experiments, and a final project played a crucial role in reinforcing theoretical knowledge. Practical implementation helped bridge the gap between conceptual understanding and real-world application, allowing for deeper insight into data preprocessing, model training, evaluation, and optimization techniques. Working with various algorithms and frameworks also enhanced problem-solving skills and technical confidence.

Overall, the WiDS program significantly strengthened my analytical thinking, programming proficiency, and understanding of modern data science workflows. The knowledge and skills gained through this program have motivated me to further explore advanced topics in data science and apply machine learning and deep learning techniques to real-world problems and future academic or professional projects.