This project gave us an excellent opportunity to understand the math that goes into and makes up machine learning and the various techniques and methodologies involved. We were given assignments which tested our understanding of the concepts and they included various proving steps so that we could derive the equations ourselves and see how it all comes together. During this project, I learnt several new concepts and topics and found myself having to research on various ideas to be able to solve questions on the same. Below are the topics which were dealt with in this project:

1. Linear Regression: Regression models describe the relationship between variables by fitting a line to the observed data. Linear regression models use a straight line, while logistic and nonlinear regression models use a curved line. Regression allows you to estimate how a dependent variable changes as the independent variable/variables change.
   Linear regression is often the first model studied because it introduces many key concepts such as loss functions, optimization, gradients, regularization, and generalization.
   We derived the expression for weights and biases and explored Maximum Likelihood Estimation for a Gaussian Model and Bayes' Rule and Classification.

2. Logistic Regression: Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for class outcomes.
   The linear regression model can work well for regression, but fails for classification. Why is that? In the case of two classes, you could label one of the classes with 0 and the other with 1 and use linear regression. Technically, it works, and most linear model programs will spit out weights for you. But there are a few problems with this approach: A linear model does not output probabilities, but it treats the classes as numbers (0 and 1) and fits the best hyperplane (for a single feature, it's a line) that minimizes the distances between the points and the hyperplane. So it simply interpolates between the points, and you cannot interpret it as probabilities.
   Instead of fitting a straight line or hyperplane, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1.
   On the good side, the logistic regression model is not only a classification model, but also gives you probabilities. This is a big advantage over models that can only provide the final classification. Knowing that an instance has a 99% probability for a class compared to 51% makes a big difference.
   Logistic regression has been widely used by many different people, but it struggles with its restrictive expressiveness (e.g., interactions must be added manually), and other models may have better predictive performance.

Another disadvantage of the logistic regression model is that the interpretation is more difficult because the interpretation of the weights is multiplicative and not additive.

Logistic regression can suffer from complete separation. If there is a feature that would perfectly separate the two classes, the logistic regression model can no longer be trained. This is because the weight for that feature would not converge, because the optimal weight would be infinite. This is really a bit unfortunate because such a feature is really useful. But you do not need machine learning if you have a simple rule that separates both classes. The problem of complete separation can be solved by introducing penalization of the weights or defining a prior probability distribution of weights.

3. Decision Trees: A decision tree is a supervised learning algorithm used for both classification and regression tasks. It has a hierarchical tree structure which consists of a root node, branches, internal nodes and leaf nodes. It works like a flowchart help to make decisions step by step where:
   - Internal nodes represent attribute tests
   - Branches represent attribute values
   - Leaf nodes represent final decisions or predictions.

   A decision tree splits the dataset based on feature values to create pure subsets, ideally all items in a group belong to the same class. Each leaf node of the tree corresponds to a class label and the internal nodes are feature-based decision points.

   A decision tree works by breaking down data step by step asking the best possible questions at each point and stopping once it reaches a clear decision. It's an easy and understandable way to make choices. Because of their simple and clear structure, decision trees are very helpful in machine learning for tasks like sorting data into categories or making predictions.

There are several disadvantages of decision trees that make them less valuable or restrict their use in many cases. Following are the most prominent disadvantages of decision trees.

- Overfitting

Among the most common and prominent disadvantages of decision trees are that it's a high variance algorithm. This means that it can easily overfit because it has no inherent mechanism to stop, thereby creating complex decision rules. As discussed before, various parameters can be tuned to control the splitting process, or pruning can be performed, but its effectiveness can be limited.

- Optimization

At every level, the decision tree algorithm looks for the pure node and doesn't consider how the recent decision will affect the next few stages of splitting. This is the reason that it is known as a greedy algorithm.

This heuristic method of working makes the model interpretable but doesn't ensure that the algorithm will return the globally optimal result. Also, if a few variables are highly significant or cause data leakage, they will 'hijack' the process. All these issues can be resolved by using an ensemble of decision trees, but that results in a complete loss of interpretability.

All these advantages and disadvantages of decision trees must be kept in mind when deciding on the algorithm to build a model as you can achieve better results by making an informed decision.

4. Random Forests: Random Forest is a machine learning algorithm that uses many decision trees to make better predictions. Each tree looks at different random parts of the data and their results are combined by voting for

classification or averaging for regression which makes it an ensemble learning technique. This helps in improving accuracy and reducing errors. **Working of Random Forest Algorithm**

- **Create Many Decision Trees:** The algorithm makes many decision trees each using a random part of the data. So every tree is a bit different.
- **Pick Random Features:** When building each tree it doesn't look at all the features (columns) at once. It picks a few at random to decide how to split the data. This helps the trees stay different from each other.
- **Each Tree Makes a Prediction:** Every tree gives its own answer or prediction based on what it learned from its part of the data.
- **Combine the Predictions:** For **classification** we choose a category as the final answer is the one that most trees agree on i.e majority voting and for **regression** we predict a number as the final answer is the average of all the trees predictions.
- **Why It Works Well:** Using random data and features for each tree helps avoid overfitting and makes the overall prediction more accurate and trustworthy.

5. Ensemble Methods: Ensemble learning is a method where we use many small models instead of just one. Each of these models may not be very strong on its own, but when we put their results together, we get a better and more accurate answer. It's like asking a group of people for advice instead of just one person—each one might be a little wrong, but together, they usually give a better answer.

Types of Ensembles Learning in Machine Learning
There are three main types of ensemble methods:

1. **Bagging (Bootstrap Aggregating):** Models are trained independently on different random subsets of the training data. Their results are then combined—usually by averaging (for regression) or voting (for classification). This helps reduce variance and prevents overfitting.

2. **Boosting:** Models are trained one after another. Each new model focuses on fixing the errors made by the previous ones. The final prediction is a weighted combination of all models, which helps reduce bias and improve accuracy.
3. **Stacking (Stacked Generalization):** Multiple different models (often of different types) are trained and their predictions are used as inputs to a final model, called a meta-model. The meta-model learns how to best combine the predictions of the base models, aiming for better performance than any individual model.

6. Confusion Matrix: Confusion matrix is a simple table used to measure how well a classification model is performing. It compares the predictions made by the model with the actual results and shows where the model was right or wrong. This helps you understand where the model is making mistakes so you can improve it. It breaks down the predictions into four categories:

- **True Positive (TP):** The model correctly predicted a positive outcome i.e the actual outcome was positive.
- **True Negative (TN):** The model correctly predicted a negative outcome i.e the actual outcome was negative.
- **False Positive (FP):** The model incorrectly predicted a positive outcome i.e the actual outcome was negative. It is also known as a Type I error.
- **False Negative (FN):** The model incorrectly predicted a negative outcome i.e the actual outcome was positive. It is also known as a Type II error.

7. Neural Networks: Neural networks are machine learning models that mimic the complex functions of the human brain. These models consist of interconnected nodes or neurons that process data, learn patterns and enable tasks such as pattern recognition and decision-making.

Neural networks are capable of learning and identifying patterns directly from data without pre-defined rules. These networks are built from several key components:

- **Neurons**: The basic units that receive inputs, each neuron is governed by a threshold and an activation function.
- **Connections**: Links between neurons that carry information, regulated by weights and biases.
- **Weights and Biases**: These parameters determine the strength and influence of connections.
- **Propagation Functions**: Mechanisms that help process and transfer data across layers of neurons.
- **Learning Rule**: The method that adjusts weights and biases over time to improve accuracy.

Learning in neural networks follows a structured, three-stage process:

1. **Input Computation: Data is fed into the network.**
2. **Output Generation: Based on the current parameters, the network generates an output.**
3. **Iterative Refinement: The network refines its output by adjusting weights and biases, gradually improving its performance on diverse tasks.**

Layers in Neural Network Architecture

1. **Input Layer:** This is where the network receives its input data. Each input neuron in the layer corresponds to a feature in the input data.
2. **Hidden Layers:** These layers perform most of the computational heavy lifting. A neural network can have one or multiple hidden layers. Each layer consists of units (neurons) that transform the inputs into something that the output layer can use.

3. **Output Layer:** The final layer produces the output of the model. The format of these outputs varies depending on the specific task like classification, regression.

Applications
Neural networks have numerous applications across various fields:

1. **Image and Video Recognition**: CNNs are extensively used in applications such as facial recognition, autonomous driving and medical image analysis.
2. **Natural Language Processing (NLP)**: RNNs and transformers power language translation, chatbots and sentiment analysis.
3. **Finance**: Predicting stock prices, fraud detection and risk management.
4. **Healthcare**: Neural networks assist in diagnosing diseases, analyzing medical images and personalizing treatment plans.
5. **Gaming and Autonomous Systems**: Neural networks enable real-time decision-making, enhancing user experience in video games and enabling autonomous systems like self-driving cars.

8. ReLU Activation Function:

Rectified Linear Unit (ReLU) is a popular activation functions used in neural networks, especially in deep learning models. It has become the default choice in many architectures due to its simplicity and efficiency. The ReLU function is a piecewise linear function that outputs the input directly if it is positive; otherwise, it outputs zero.

In simpler terms, ReLU allows positive values to pass through unchanged while setting all negative values to zero. This helps the neural network maintain the necessary complexity to learn patterns while avoiding some of the pitfalls associated with other activation functions

Why is ReLU Popular?

- **Simplicity:** ReLU is computationally efficient as it involves only a thresholding operation. This simplicity makes it easy to implement and compute, which is important when training deep neural networks with millions of parameters.

- **Non-Linearity:** Although it seems like a piecewise linear function, ReLU is still a non-linear function. This allows the model to learn more complex data patterns and model intricate relationships between features.
- **Sparse Activation:** ReLU's ability to output zero for negative inputs introduces sparsity in the network, meaning that only a fraction of neurons activate at any given time. This can lead to more efficient and faster computation.
- **Gradient Computation:** ReLU offers computational advantages in terms of backpropagation, as its derivative is simple—either 0 (when the input is negative) or 1 (when the input is positive). This helps to avoid the vanishing gradient problem, which is a common issue with sigmoid or tanh activation functions.

Finally, I am grateful for the mentors’ interest and detailed assignments and resources provided to facilitate our learning and hope to explore more in the future.