Summer of Science 2024

AIML

Manit Jhajharia (23B1265)

https://github.com/Osama-Bin-Lagging/SOS-2024—AIML

Contents

1. Introduction to Machine Learning

1.1 Objectives

- Understand the fundamental concepts of supervised and unsupervised learning.
- Learn the basics of linear regression, logistic regression, and key concepts like overfitting, underfitting, and model evaluation.
- Get introduced to clustering methods (K-means) and dimensionality reduction techniques (PCA).

1.2 Overview

Machine Learning (ML) is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to "learn" from data without being explicitly programmed. This week is focused on building a strong foundation in understanding the basic principles of machine learning, which will pave the way for more complex topics in the following weeks.

1.3 Detailed Topics

1.3.1 Supervised Learning

- **Definition** Learning from a labeled dataset, where the outcome (or label) is known. The model learns to predict the output from the input data.
- **Examples** Linear regression, logistic regression, decision trees, etc.

1.3.2 Unsupervised Learning

- **Definition** Learning from an unlabeled dataset, where the model tries to identify patterns and structures in the data without predefined labels.
- **Examples** Clustering (K-means, hierarchical clustering), dimensionality reduction (PCA, t-SNE).

1.3.3 Linear Regression

- **Purpose** Predict a continuous target variable based on one or more predictor variables.
- **Mathematical Model** $Y = \beta_0 + \beta_1 X$
- **Use Case** Predicting house prices based on square footage, number of bedrooms, etc.

1.3.4 Logistic Regression

- **Purpose** Predict a binary outcome (0 or 1) from a set of predictor variables.
- **Mathematical Model** $P(Y = 1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$
- **Use Case** Determining whether an email is spam or not.

1.3.5 Key Concepts

- **Overfitting** A model learns the noise in the training data, performing well on training data but poorly on new data.
- **Underfitting** A model is too simple to capture the underlying trend in the data, resulting in poor performance on both training and new data.
- **Model Evaluation** Techniques to assess a model's performance, such as cross-validation, accuracy, precision, recall, and F1 score.

1.3.6 Clustering (K-means)

- **Purpose** Group data points into clusters based on their similarities.
- **Algorithm Steps**
  1. Choose the number of clusters (k).
  2. Randomly initialize k centroids.
  3. Assign each data point to the nearest centroid.
  4. Update the centroids based on the assigned data points.
  5. Repeat steps 3 and 4 until convergence.

1.3.7 Dimensionality Reduction (PCA)

- **Purpose** Reduce the number of variables in the dataset while preserving as much information as possible.
- **Algorithm Steps**
  1. Standardize the data.
  2. Compute the covariance matrix.
  3. Calculate the eigenvalues and eigenvectors.

4. Select the top k eigenvectors (principal components).
5. Transform the data into the new subspace.

1.4 Activities

- **Lectures** Watched the first two weeks of the "Machine Learning" course by Andrew Ng on Coursera. This covered the basics of linear and logistic regression and introduce overfitting and underfitting concepts.
- **Books** Read the first Chapter of Hands On Machine Learning with Scikit Learn, Keras and TensorFlow, 2nd-Edition by Aurelien Geron
- **Practical Implementation**
  - Implement linear regression on a dataset (e.g., predicting house prices).
  - Implement K-means clustering on a sample dataset (e.g., customer segmentation).
  - Use Python libraries like Scikit-Learn for these implementations.

Resources

- "Machine Learning" by Andrew Ng on Coursera
- Hands On Machine Learning with Scikit Learn, Keras and TensorFlow, 2nd-Edition by Aurelien Geron

2. Supervised Learning Algorithms

2.1 Objectives

- Dive deeper into supervised learning algorithms for both classification and regression tasks.
- Learn and implement decision trees, random forests, and support vector machines (SVM) for classification.
- Understand and apply polynomial regression and regularization techniques (Ridge, Lasso).
- Use Scikit-Learn to implement and compare these algorithms on datasets like the Iris dataset.

2.2 Overview

This week focuses on enhancing your understanding of various supervised learning algorithms, which are critical for solving both classification and regression problems. You will learn how to implement these algorithms and compare their performance on different datasets.

2.3 Detailed Topics

2.3.1 Classification Algorithms

- **Decision Trees (Implemented)**
  - **Definition** A tree-like model where each internal node represents a decision based on a feature, and each leaf node represents an outcome.
  - **Advantages** Easy to understand and interpret, can handle both numerical and categorical data.

  - **Disadvantages** Prone to overfitting.
- **Random Forests (Implemented)**
  - **Definition** An ensemble method that creates a 'forest' of decision trees and merges their results to improve accuracy and control overfitting.
  - **Advantages** High accuracy, less overfitting, handles large datasets well.
  - **Disadvantages** Less interpretable than a single decision tree.
- **Support Vector Machines (SVM) (Not Implemented)**
  - **Definition** A classifier that finds the hyperplane that best separates the data into classes.
  - **Advantages** Effective in high-dimensional spaces, robust to overfitting.
  - **Disadvantages** Requires careful tuning of parameters, not very effective on large datasets.

2.4 Regression Algorithms

(Not Implemented)

- **Polynomial Regression**
  - **Definition** A form of regression that models the relationship between the dependent variable and the independent variable as an n-th degree polynomial.
  - **Use Case** Capturing non-linear relationships.
- **Regularization Techniques**
  - **Ridge Regression** Adds a penalty equivalent to the square of the magnitude of coefficients to the loss function.
  - **Lasso Regression** Adds a penalty equivalent to the absolute value of
    the magnitude of coefficients to the loss function.
  - **Purpose** Prevent overfitting by penalizing large coefficients.

2.5 Activities

- **Lectures** Watch tutorials on decision trees, random forests, SVMs, polynomial regression, and regularization techniques.
- **Books** Read the second Chapter of Hands On Machine Learning with Scikit Learn, Keras and TensorFlow, 2nd-Edition by Aurelien Geron
- **Practical Implementation**
  - Use a to implement decision trees, random forests, and SVM for regression (predicting housing prices).
  - Compare the performance of different algorithms using Scikit-Learn.

Resources

- Scikit-Learn documentation for classification and regression algorithms

- Hands On Machine Learning with Scikit Learn, Keras and TensorFlow, 2nd-Edition by Aurelien Geron

2.6 Implementation

Predicting Housing Prices, and applying data different analysis tools to study the data

2.6.1 Data Analysis

1. Taking a look at the data we have,

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Figure 2.1: Top five rows in the dataset

The Dataset contains information of blocks of California, namely their longitude, latitude, housing median age, total rooms, total bedrooms, population, households, median income, median house value and ocean proximity. (Note that all data except ocean proximity are numerical, which would be converted to numbers later on)

2. The information about each data column is as follows:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

Figure 2.2: A histogram for each numerical attribute

3. Another quick way to at the type of data we are dealing with is to plot a histogram for each numerical attribute

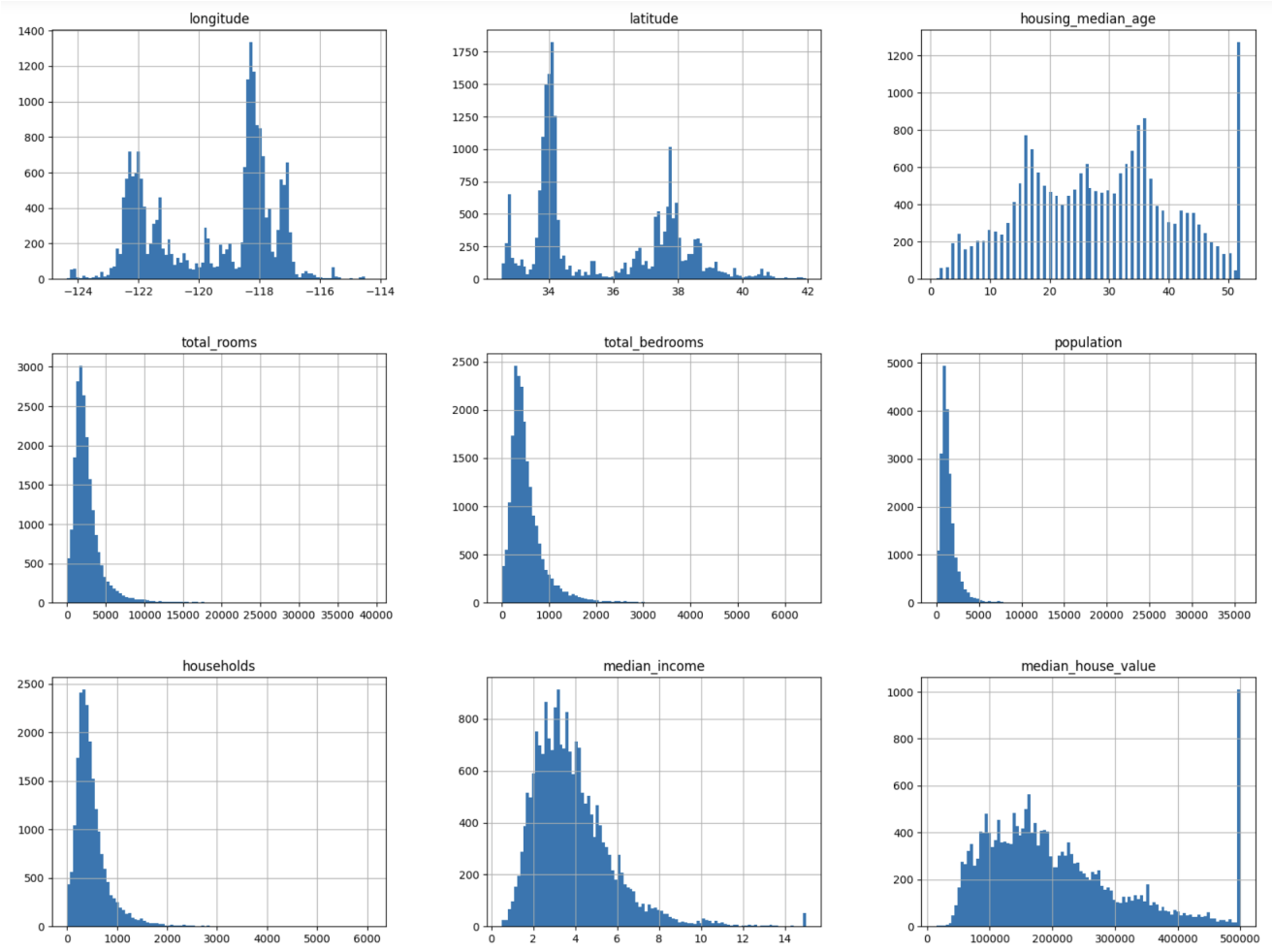

Figure 2.3: A histogram for each numerical attribute

4. Visualizing Geographical Data

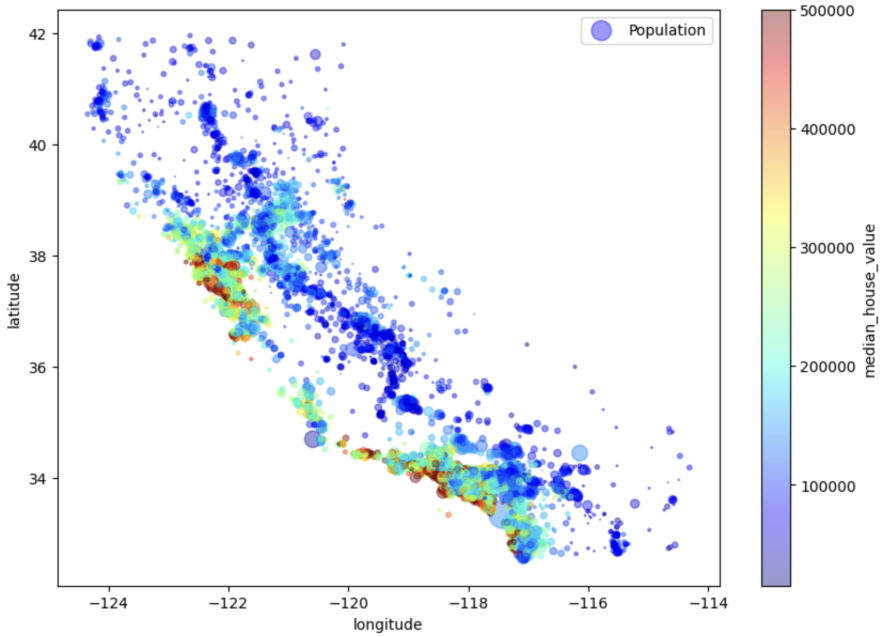

Figure 2.4: California housing prices: red is expensive, blue is cheap, larger circles indi- cate areas with a larger population

2.6.2 Trying Models

1. Using Linear Regression, we get a root mean squared error(RMSE) of 68633.40810776998 and an accuracy of 65.0%

2. Using Decision Tree Regression, we get a root mean squared error of 0 and an accuracy of 100%, but using Scikit-Learn's K-fold cross-validation feature, on running a 10 fold cross validation, we get a mean RMSE of 69116.4347200802, while for the same for a Linear Regression model is 50291.390346787455. We see that Decision Tree Regressor highly overfits the data.

3. Using Random Forest Regression, we get a root mean squared error of 18696.182706622574 and an accuracy of 97.0%. The 10 fold cross-validation score for the Random Forest Regressor is 18696.182706622574. However, the score on the training set is still much lower than on the validation sets, meaning that the model is still overfitting the training set.

2.6.3 Fine-tuning the model

1. Manually setting the parameters using Grid Search CV. All we need to do is tell it which hyperparameters we want it to experiment with and what values to try out, and it will use cross-validation to evaluate all

the possible combinations of hyperparameter values. We can directly get the best parameters, best estimator and individual parameter evaluation scores.

2. The grid search approach is fine when you are exploring relatively few combinations, but when the hyperparameter search space is large, it is often preferable to use Randomized Search CV instead.

2.6.4 Final Prediction

We get a final accuracy of 83.0% with a RMSE of 46519.46106514513

3. Advanced Classification Techniques

3.1 Objectives

- Train a binary classifier on the MNIST dataset.
- Understand and implement various performance measures.
- Learn how to measure accuracy using cross-validation.
- Explore confusion matrix, precision, recall, and the precision/recall trade-off.
- Study the Receiver Operating Characteristic (ROC) curve.
- Dive into multiclass classification, error analysis, multilabel classification, and multioutput classification.

3.2 Overview

This week focused on advanced classification techniques and performance evaluation metrics. I train a binary classifier on the MNIST dataset, delved into various metrics to assess model performance, and explore multiclass and multilabel classification.

3.3 Detailed Topics

3.3.1 MNIST Dataset:

- **Description:** The MNIST dataset consists of 70,000 images of handwritten digits (0-9) in grayscale, each with a resolution of 28x28 pixels.
- **Use Case:** A benchmark dataset for training and testing image processing systems. (Also known as the "print('hello world')" of AIML community)

3.3.2 Training a Binary Classifier:

- **Objective:** Build a binary classifier to distinguish between two classes (e.g., digit 5 vs. not 5).
- **Steps:**
  1. Load and preprocess the MNIST dataset.
  2. Create binary labels for the target class (e.g., label 5 as positive and all others as negative).
  3. Train the classifier using algorithms like Logistic Regression or Support Vector Machine (SVM).

3.3.3 Performance Measures:

- **Accuracy:** The ratio of correctly predicted instances to the total instances.
- **Cross-Validation:** A technique to evaluate model performance by dividing data into k subsets and training/testing the model k times.

3.3.4 Measuring Accuracy Using Cross-Validation:

- **Purpose:** Provides a better estimate of the model's performance by testing it on multiple data splits.
- **Steps:**
  1. Split the data into k folds.
  2. Train the model on k-1 folds and test on the remaining fold.
  3. Repeat the process k times and average the results.

3.3.5 Confusion Matrix:

- **Description:** A table used to describe the performance of a classification model by comparing actual and predicted values.
- **Components:**
  - **True Positives (TP):** Correctly predicted positive instances.
  - **True Negatives (TN):** Correctly predicted negative instances.
  - **False Positives (FP):** Incorrectly predicted positive instances.
  - **False Negatives (FN):** Incorrectly predicted negative instances.

3.3.6 Precision and Recall:

- **Precision:** The ratio of correctly predicted positive instances to the total predicted positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** The ratio of correctly predicted positive instances to the total actual positives.

$$\text{Recall} = \frac{TP}{TP + FN}$$

3.3.7 Precision/Recall Trade-off:

- **Explanation:** Increasing precision typically reduces recall and vice versa.
- **Purpose:** To find a balance based on the problem's requirements (e.g., high recall for identifying all relevant cases).

3.3.8 The ROC Curve:

- **Description:** A graphical plot illustrating the diagnostic ability of a binary classifier as its discrimination threshold is varied.
- **Components:**
  - **True Positive Rate (TPR):** Also known as recall or sensitivity.
  - **False Positive Rate (FPR):** The ratio of falsely predicted positive instances to the total actual negatives.
- **AUC (Area Under Curve):** A measure of the model's ability to distinguish between classes.

3.3.9 Multiclass Classification:

- **Description:** Extends binary classification to multiple classes (e.g., recognizing all digits 0-9 in MNIST).
- **Approaches:**
  - **One-vs-All (OvA):** Train one classifier per class, treating it as a binary problem. (Implemented)
  - **One-vs-One (OvO):** Train a classifier for every pair of classes. (Not Implemented)

3.3.10 Error Analysis:

- **Purpose:** To diagnose and understand errors made by the model.
- **Process:**
  1. Analyze the confusion matrix to identify common misclassifications.
  2. Examine misclassified instances to detect patterns or sources of error.

Multilabel Classification:

- **Description:** A classification task where each instance can belong to multiple classes simultaneously.
- **Use Case:** Tagging images with multiple objects (e.g., identifying both cats and dogs in an image).

3.3.11 Multioutput Classification:

- **Description:** A generalization of multilabel classification where multiple labels are predicted, each corresponding to a different output.
- **Use Case:** Predicting multiple labels for different aspects of a single instance (e.g., predicting multiple attributes of a house).

3.4 Activities

- **Lectures:** Watch tutorials and lectures on advanced classification techniques and evaluation metrics.
- **Practical Implementation:**
  - Use the MNIST dataset to train a binary classifier (e.g., classifying digit 5 vs. not 5).
  - Implement cross-validation to measure model accuracy.
  - Construct and analyze a confusion matrix.
  - Calculate precision and recall, and explore the precision/recall trade-off.
  - Plot and interpret the ROC curve.
  - Implement multiclass classification on the MNIST dataset.
  - Perform error analysis on the classifier.
  - Implement multilabel and multioutput classification on relevant datasets.

3.5 Resources

- Scikit-Learn documentation for classification and evaluation metrics
- "Machine Learning" by Andrew Ng on Coursera
- Hands On Machine Learning with Scikit Learn, Keras and TensorFlow, 2nd-Edition by Aurelien Geron
- MNIST dataset available through Scikit-Learn

3.6 Implementation

3.6.1 Binary Classifier

1. We have a data of 70,000 images, each with a 28×28 pixel size, each pixel corresponding to a binary value. Thus the data points consists of a (70000,784) matrix with a (70000,1) matrix of corresponding labels. an example image is shown below

Figure 3.1: The image corresponding to the 0th index

2. First I worked on a simple binary classifer which classifies a number a 5 or not-5. Creating our own version of the 3 fold cross validation score on a SGD Classifier, we predict an accuracy of $[0.9521, 0.96445, 0.9613]$, while using Scikit-Learn's internal function, we get an accuracy of $[0.9553, 0.9441, 0.95595]$, which is quite close

3. The Confusion Matrix is a good tool which tells us how much we have miss-classified the data. The confusion matrix of the SGD Classifier used is $\begin{bmatrix} 53022 & 1557 \\ 969 & 4452 \end{bmatrix}$ Each row in a confusion matrix represents an actual class, while each column represents a predicted class.

4. Calculating precision and recall from the confusion matrix we get them as 0.7408886669995007 and 0.8212506917542889, respectively. The F1 score is the harmonic mean of precision and recall and combines precision and recall into a single metric. The value of F1 score is 0.779002624671916

5. We plot precision and recall vs decision threshold to see how the precision and recall changes with threshold value.

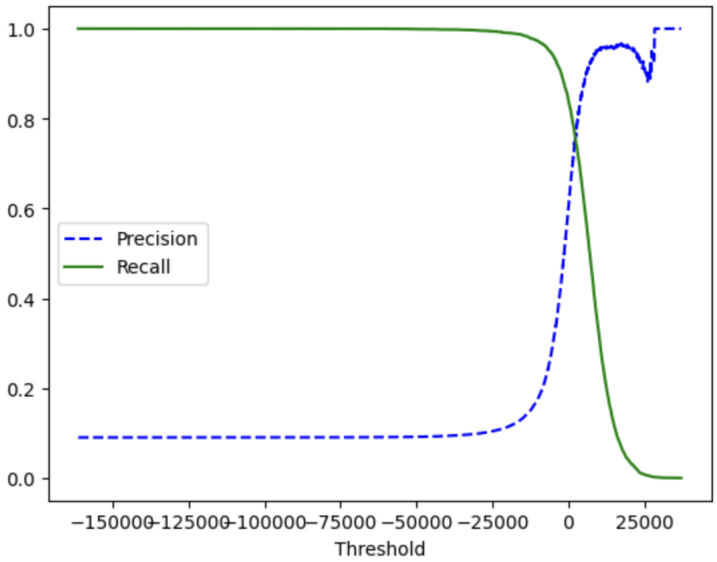

Figure 3.2: Precision and recall versus the decision threshold

precision may sometimes go down when you raise the threshold (although in general it will go up). On the other hand, recall can only go down when the threshold is increased, which explains why its curve looks smooth.

6. Another way to select a good precision/recall trade-off is to plot precision directly against recall.

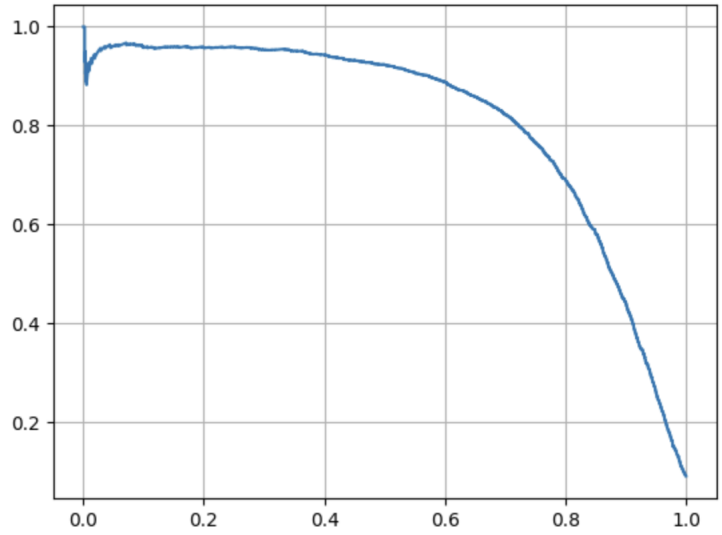

Figure 3.3: Precision versus recall

7. The higher the recall (TPR), the more false positives (FPR) the classifier produces. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible. The Area Under Curve is (0.9531738134770292)

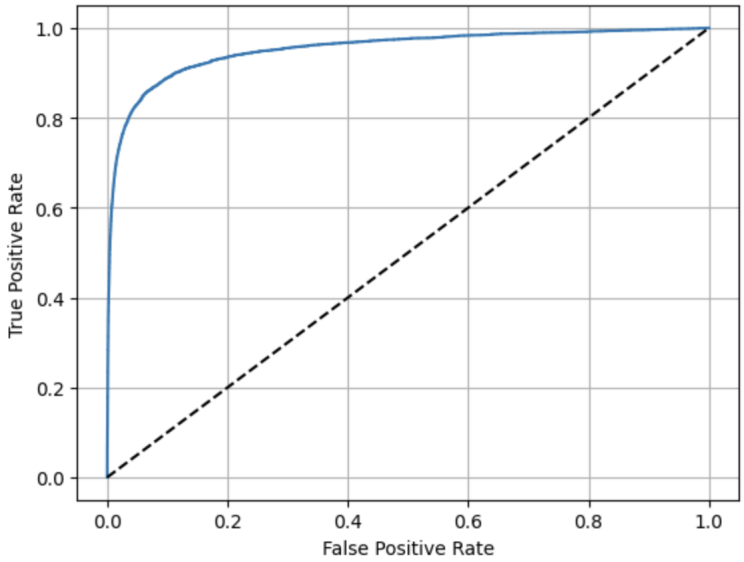

Figure 3.4: This ROC curve plots the false positive rate against the true positive rate for all possible thresholds

8. Now on train a Random Forest Classifier and we plot its ROC curve and get a ROC AUC of 0.9981834910999697, which is considerably higher.

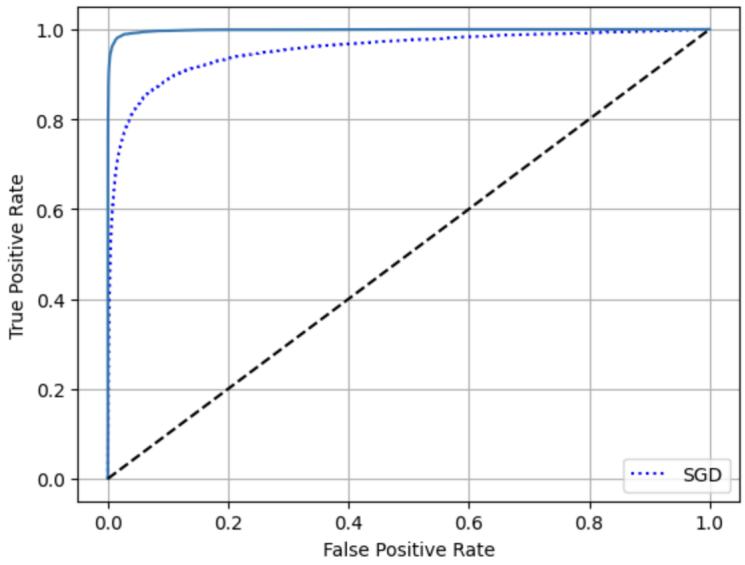

Figure 3.5: the Random Forest classifier is superior to the SGD classifier because its ROC curve is much closer to the top-left corner, and it has a greater AUC

3.6.2 Multiclass Classification

1. One strategy is to train a binary classifier for every pair of digits: one to distinguish 0s and 1s, another to distinguish 0s and 2s, another for 1s and 2s, and so on. This is called the one-versus-one (OvO) strategy. The SVC is one such Classifier. The 3 fold cross validation score reveals a score of (0.8784, 0.8865, 0.8544), all are above 85%.
   If you used a random classifier, you would get 10% accuracy, so this is not such a bad score.

2. Error Analysis
   - After normalizing the confusion matrix for multiclass classification, we get a plot shown in Figure 3.6
     The column for class 8 is quite bright, which tells you that many images get misclassified as 8s. However, the row for class 8 is not that bad, telling you that actual 8s in general get properly classified as 8s.

3.6.3 Multilabel Classification

1. In some cases you may want your classifier to output multiple classes for each instance. Implementing a K Nearest Neighbour (KNN) Classifier for 2 categories, number larger than 7 and number being odd, we get a F1 score of 0.9764102655606048

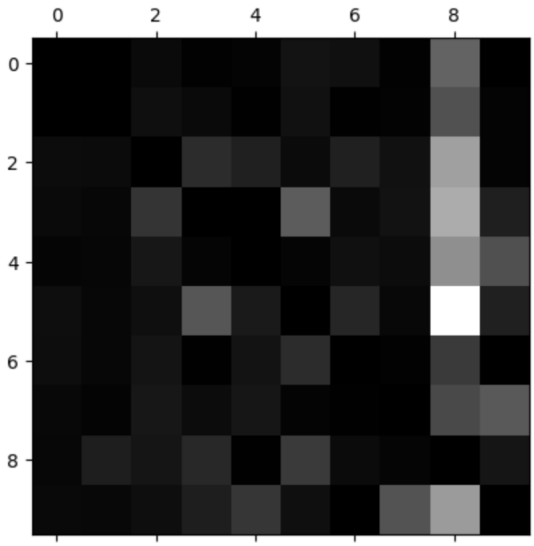

Figure 3.6: Rows represent actual classes, while columns represent predicted classes.

4. Neural Networks and Deep Learning

4.1 Objectives

- Understand the basics of neural networks, including neurons, activation functions, and the processes of forward and backward propagation.
- Get introduced to popular deep learning frameworks like TensorFlow and Keras.
- Build a simple neural network for digit recognition using the MNIST dataset.

4.2 Overview

Deep learning, a subset of machine learning, involves neural networks with multiple layers that can model complex relationships in data. This week is focused on understanding the foundational concepts of neural networks and gaining hands-on experience with deep learning frameworks.

4.3 Detailed Topics

4.3.1 Neural Networks

- **Neurons** The basic unit of a neural network, mimicking the human brain's neurons, which receives input, processes it, and passes it to the next layer.
- **Activation Functions** Functions applied to neurons to introduce non-linearity. Common activation functions include:
  - **ReLU (Rectified Linear Unit)** $f(x) = \max(0, x)$
  - **Sigmoid** $f(x) = \frac{1}{1+e^{-x}}$
  - **Tanh** $f(x) = \tanh(x)$

4.3.2 Forward and Backward Propagation

- **Forward Propagation** The process of passing input data through the neural network to get the output.
- **Backward Propagation** The process of adjusting the weights in the network by propagating the error back through the network to minimize the loss function.

4.3.3 Deep Learning Frameworks

- **TensorFlow** An open-source library for numerical computation and machine learning.
- **Keras** A high-level API for building and training deep learning models, often used with TensorFlow as the backend.

4.4 Activities

- **Lectures** Watch tutorials and lectures on neural networks and deep learning frameworks.
- **Practical Implementation**
  - Use TensorFlow and build and train a simple neural network for recogniting pizza.
  - experimenting with different activation functions to see their impact on model performance.

4.5 Implementation

1. The images are 64 $\times$ 64 pixelated colour images, meaning each pixel in turn is represented as an array of RGB values

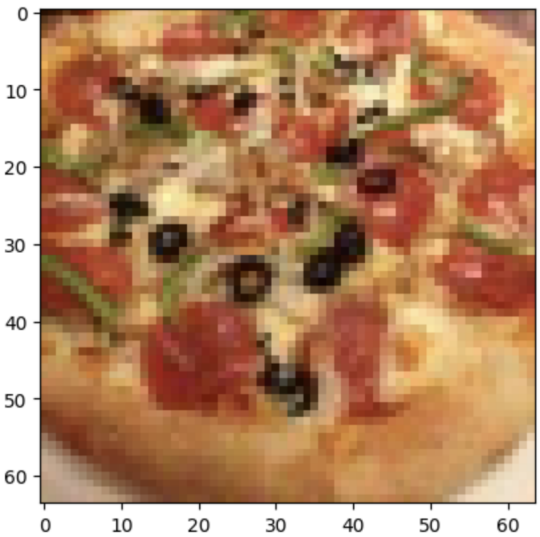

Figure 4.1: Example Image

2. After trial and error we get a model with decent accuracy of a 59.0%. Here, I made a 3 layer neural network (i.e., 1 input layer, 2 hidden layers and 1 output layer), the first hidden layer with 10 nodes, while the second with 12.
3. The Model includes the use of Linear function at each layer, alongside non-linear activation functions.

4. We used a ReLU functions as the activation function for the input and the first hidden layer and Sigmoid function for the final output, as we want an output between 0 and 1.