Name: Ojei Victory

Student Number: 202137995

Course: MSc Artificial Intelligence & Data Science

Module Name: UNDERSTANDING AI MODULE 771763_A24_T1

Contents

Introduction

This project is divided into three components, each highlighting distinct applications of machine learning. Component one focuses on predicting global video game sales using regression models such as linear, polynomial, and artificial neural networks (ANNs). It also applies clustering algorithms like K-Means and DBSCAN to uncover hidden patterns in the data. Key features such as regional sales, critic scores, and user engagement are integrated to enhance predictive accuracy, with evaluation metrics like Mean Squared Error (MSE), R^2, and silhouette scores assessing model and clustering performance. Component two develops a Convolutional Neural Network (CNN) to classify emergency and non-emergency vehicles. The study explores architectural designs, regularization techniques, and hyperparameter tuning to achieve optimal performance while addressing challenges like class imbalance. Component three critically examines ethical considerations in artificial intelligence, emphasizing transparency, fairness, and accountability. Together, these components demonstrate practical machine learning applications and promote ethical AI practices in real-world scenarios.

COMPONENT ONE: Sales Performance of Video Games

Single Numerical Input Feature for Regression Model

This aspect focuses on using supervised regression models to predict the global sales of video games based on single numerical input features. The task begins by evaluating important numerical variables from the dataset provided, such as regional sales (e.g., NA_Sales, EU_Sales, JP_Sales, Other_Sales) and critic/user-related metrics (e.g., Critic_Score, User_Count), to identify the best predictors of global sales. These numerical variables were chosen because, regional sales should contribute directly to global sales, as global sales are likely the sum of these values from the dataset. These features are likely to show strong correlations with the target variable and serve as logical predictors. For Critic_Score, reviews and ratings/feedback by critics are often a significant factor influencing the success of a video game. High critic scores can drive higher sales due to increased consumer trust and positive publicity. User_Count is the number of users who reviewed or rated the game. A high user count may indicate greater popularity or awareness, which could be indirectly correlated with higher sales.

Both linear and non-linear models (polynomial regression), are applied to assess the relationship between each feature and global sales. For each feature, performance metrics such as Mean Squared Error (MSE) and R^2 are computed to evaluate model accuracy and fit. The analysis also involves visualizing the data and regression curves to highlight the predictive strength and trend of each variable.

The results provided show that regional sales variables, particularly NA_Sales, demonstrate the highest correlation with global sales and achieve superior predictive performance compared to other numerical features. Polynomial regression often improves fit slightly over linear regression, capturing non-linear relationships where present.

This subcomponent emphasizes the importance of feature selection and model evaluation in sales forecasting. It provides foundational insights into the predictive ability of numerical features in the dataset, serving as a baseline for more complex models and multivariate analyses explored in subsequent sections of the project. This process demonstrates rigorous data-driven methodology in a real-world context.

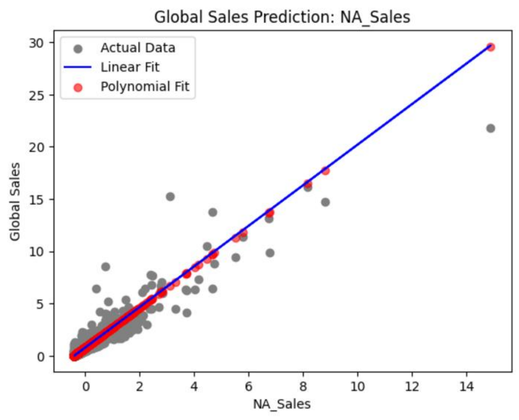


Figure 1 Visualization of NA_Sales against Global Sales

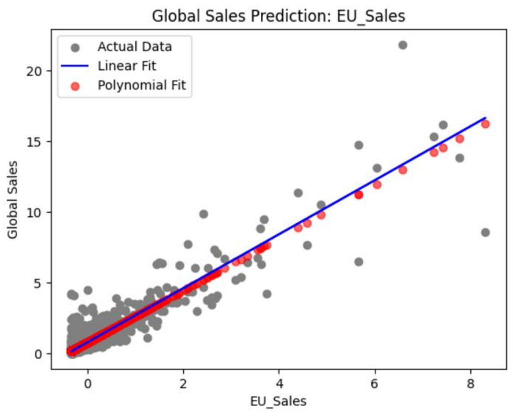


Figure 2 Visualization of EU_Sales against Global Sales

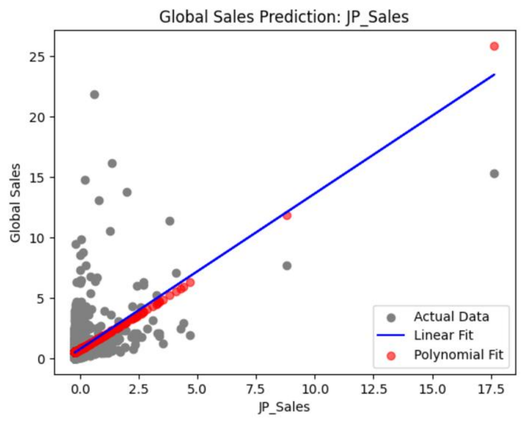


Figure 3 Visualization of JP_Sales against Global Sales

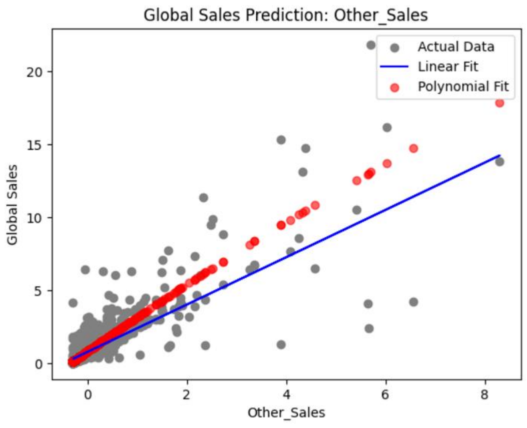


Figure 4 Visualization of Other_Sales against Global Sales

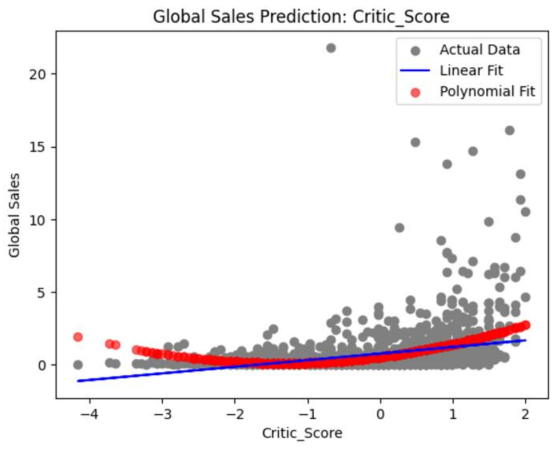


Figure 5 Visualization of Critic_Score against Global Sales

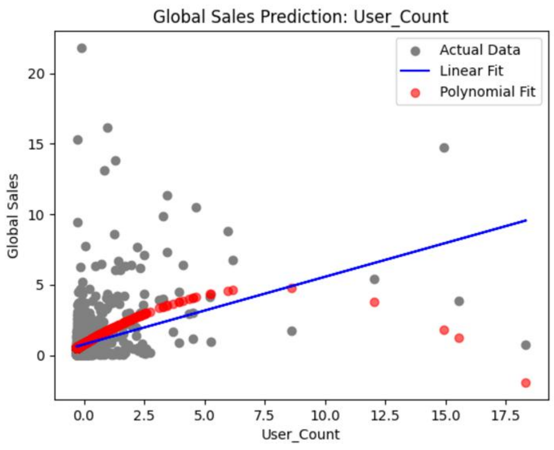
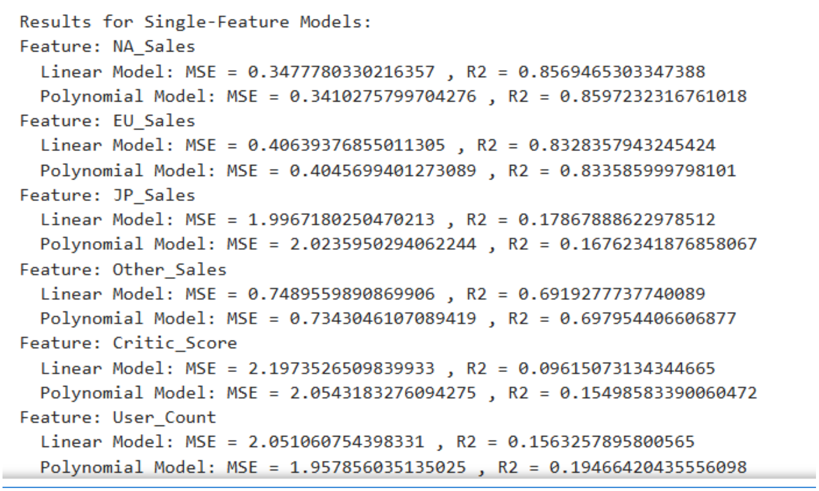


Figure 6 Visualization of User_Count against Global Sales



```
Results for Single-Feature Models:
Feature: NA_Sales
  Linear Model: MSE = 0.3477780330216357 , R2 = 0.8569465303347388
  Polynomial Model: MSE = 0.3410275799704276 , R2 = 0.8597232316761018
Feature: EU_Sales
  Linear Model: MSE = 0.40639376855011305 , R2 = 0.8328357943245424
  Polynomial Model: MSE = 0.4045699401273089 , R2 = 0.833585999798101
Feature: JP_Sales
  Linear Model: MSE = 1.9967180250470213 , R2 = 0.17867888622978512
  Polynomial Model: MSE = 2.0235950294062244 , R2 = 0.16762341876858067
Feature: Other_Sales
  Linear Model: MSE = 0.7489559890869906 , R2 = 0.6919277737740089
  Polynomial Model: MSE = 0.7343046107089419 , R2 = 0.697954406606877
Feature: Critic_Score
  Linear Model: MSE = 2.1973526509839933 , R2 = 0.09615073134344665
  Polynomial Model: MSE = 2.0543183276094275 , R2 = 0.15498583390060472
Feature: User_Count
  Linear Model: MSE = 2.051060754398331 , R2 = 0.1563257895800565
  Polynomial Model: MSE = 1.957856035135025 , R2 = 0.19466420435556098
```

Figure 7 Results for Single-Feature Models

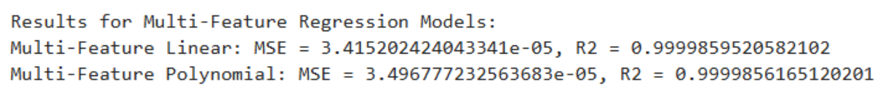
Multiple Numerical Variable for Regression Model

This task explores the use of multiple numerical input features to predict the global sales of video games and evaluates whether multivariate regression models improve prediction accuracy compared to single-feature models. This analysis integrates all numerical variables, including regional sales (e.g., NA_Sales, EU_Sales, JP_Sales, Other_Sales), Critic_Score, and User_Count, into an extensive predictive framework.

Using both linear regression and polynomial regression, the analysis evaluates the combined impact of these features on global sales. The dataset is preprocessed to address missing values and ensure numerical features are scaled for consistency. Models are trained and evaluated using metrics such as Mean Squared Error (MSE) and R^2 scores to measure their accuracy and explanatory power. The results are then compared to the performance of single-feature models analysed earlier.

The findings show that incorporating multiple features significantly improves model accuracy, as evidenced by higher R^2 scores and lower MSE values. Multivariate models capture more complex interactions between variables, leading to enhanced predictive performance. Polynomial regression provides further improvements by modelling non-linear relationships between the input features and global sales.

This subcomponent highlights the advantages of multivariate regression in leveraging multiple data dimensions to improve predictions. It underscores the importance of feature integration and advanced modelling techniques in data science-driven forecasting.



```
Results for Multi-Feature Regression Models:
Multi-Feature Linear: MSE = 3.415202424043341e-05, R2 = 0.9999859520582102
Multi-Feature Polynomial: MSE = 3.496777232563683e-05, R2 = 0.9999856165120201
```

Figure 8 Results for Multi-Feature Model

Categorical and Numerical Variable for Regression Model

This task investigates the impact of combining categorical variables alongside numerical features in regression models to predict global video game sales. This analysis acknowledges the potential influence of non-numerical attributes, such as platform, genre, publisher, and rating, on sales performance. By encoding these categorical variables and combining them with numerical features, a more comprehensive predictive framework is created.

A Random Forest Regressor is employed due to its capability to handle both categorical and numerical data effectively. The model pipeline includes preprocessing steps such as scaling numerical variables and one-hot encoding categorical features to ensure compatibility with the regression algorithm. The model is trained and evaluated using Mean Squared Error (MSE) and R^2 scores, which are compared to the results from models that only use numerical inputs.

The results demonstrate that the inclusion of categorical variables improves the model's predictive accuracy, as evidenced by lower MSE values and higher R^2 scores. This indicates that categorical factors significantly contribute to explaining variations in global sales, highlighting their importance in video game performance analysis.

This subcomponent emphasizes the value of incorporating diverse data types in predictive modelling, showcasing the effectiveness of Random Forest in handling mixed data and improving prediction reliability.

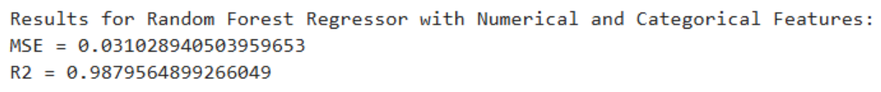

```
Results for Random Forest Regressor with Numerical and Categorical Features:
MSE = 0.031028940503959653
R2 = 0.9879564899266049
```

Figure 9 Results for Random Forest Regressor

Artificial Neural Network (ANN)

This aspect focuses on developing an Artificial Neural Network (ANN) to predict global video game sales using all relevant features in the dataset, including numerical and categorical variables. The ANN model is constructed to leverage its ability to learn complex, non-linear relationships in the data, offering a robust alternative to traditional regression models.

The architecture of the ANN includes multiple hidden layers with ReLU activation functions, dropout layers to prevent overfitting, and a single output neuron for regression. The input features are preprocessed by scaling numerical data and encoding categorical variables to ensure compatibility with the neural network. Hyperparameters such as learning rate, batch size, number of neurons, and dropout rate are tuned to optimize model performance. The model is trained using the Adam optimizer and evaluated using Mean Squared Error (MSE) and R^2 scores.

The ANN shows strong predictive performance, particularly in capturing complex interactions between features, and achieves competitive accuracy compared to traditional models like Random Forest. However, the training process is computationally intensive and requires careful tuning to avoid overfitting.

This subcomponent highlights the potential of ANNs in predictive modelling, showcasing their adaptability and ability in handling diverse datasets with complex relationships between features.

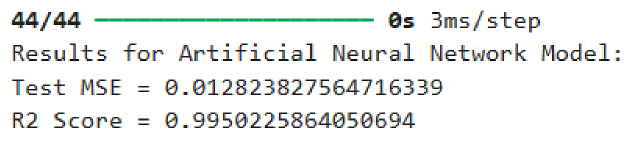
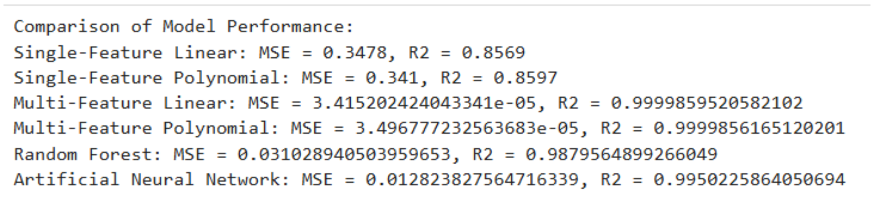

```
44/44 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Results for Artificial Neural Network Model:
Test MSE = 0.012823827564716339
R2 Score = 0.9950225864050694
```

Figure 10 Result for ANN Model

Result of Analysis/Best Model

This task evaluates the performance of various models—single feature regression, multiple feature regression, Random Forest, and Artificial Neural Networks (ANNs)—to identify the best method for predicting global video game sales. Every model is assessed using evaluation metrics such as Mean Squared Error (MSE) and R^2 scores. The results show that multivariate models outperform single feature models by exploring additional data dimensions, while Random Forest and ANNs provide further accuracy improvements due to their ability to capture complex feature relationships. The analysis concludes that ANNs produces the most reliable predictions, emphasizing the importance of advanced modelling techniques in sales prediction.



```
Comparison of Model Performance:
Single-Feature Linear: MSE = 0.3478, R2 = 0.8569
Single-Feature Polynomial: MSE = 0.341, R2 = 0.8597
Multi-Feature Linear: MSE = 3.415202424043341e-05, R2 = 0.9999859520582102
Multi-Feature Polynomial: MSE = 3.496777232563683e-05, R2 = 0.9999856165120201
Random Forest: MSE = 0.031028940503959653, R2 = 0.9879564899266049
Artificial Neural Network: MSE = 0.012823827564716339, R2 = 0.9950225864050694
```

Figure 11 Comparison of Model Performance

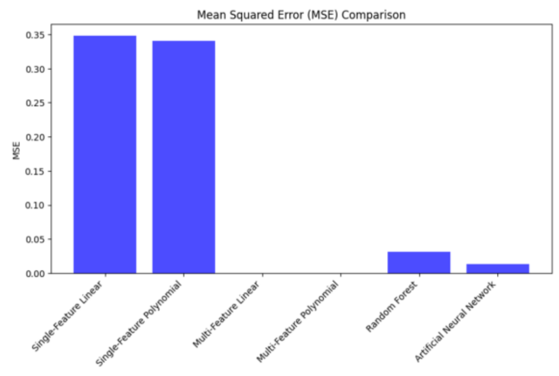


Figure 12 MSE Comparison

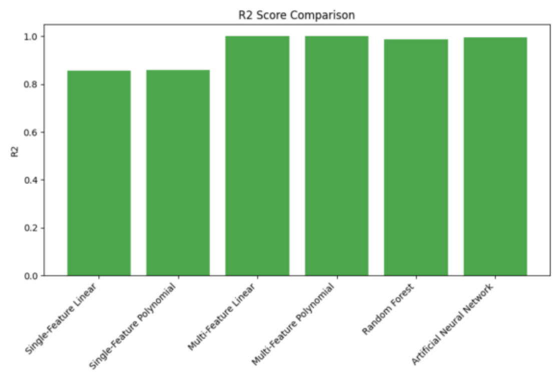
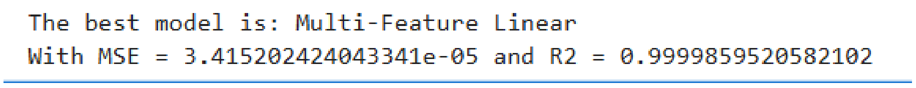


Figure 13 R^2 Comparison



```
The best model is: Multi-Feature Linear
With MSE = 3.415202424043341e-05 and R2 = 0.9999859520582102
```

Figure 14 Best Model

K-Means Clustering Algorithm

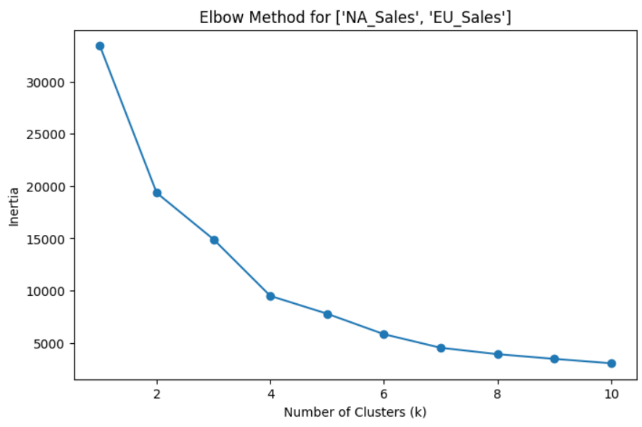
In this analysis, the goal was to perform clustering on video game sales data using numerical features to identify meaningful patterns. Specifically, K-Means clustering was applied to selected feature combinations: ['NA_Sales', 'EU_Sales'], ['JP_Sales', 'Global_Sales'], and ['NA_Sales', 'EU_Sales', 'Global_Sales']. The optimal number of clusters (k) for each combination was determined using the Elbow Method, which plots the inertia against various values of k. The "elbow" point, where the decrease in inertia slows down, was identified as the optimal k.

For all three feature combinations, the Elbow Method selected an optimal k of 4 from the graph drawn. Clustering results were evaluated using the Silhouette Score and the Davies-Bouldin Index. The Silhouette Score assesses the compactness and separation of clusters, with higher values indicating better-defined clusters, while the Davies-Bouldin Index evaluates the ratio of intra-cluster to inter-cluster distances, where lower values are preferable.

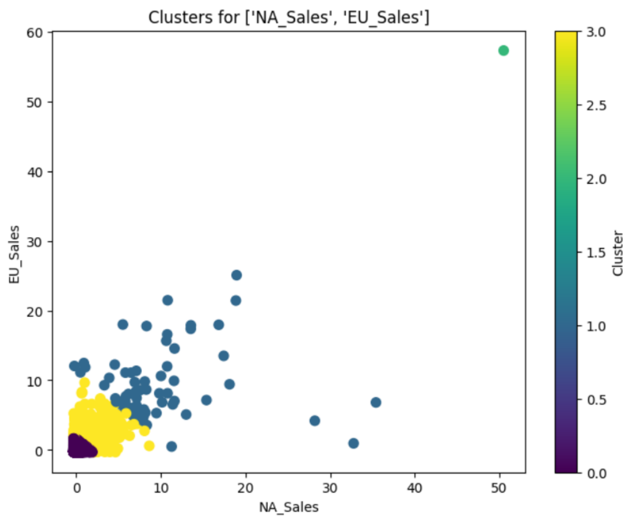
The combination ['NA_Sales', 'EU_Sales', 'Global_Sales'] yielded the best results, achieving a Silhouette Score of 0.8490 and a Davies-Bouldin Index of 0.5028, indicating well-separated and compact clusters. Comparatively, the other combinations exhibited slightly lower clustering quality. Visualizations of the clustering results further illustrated the separation and distribution of data points across the clusters.

In conclusion, the analysis successfully demonstrated the use of clustering algorithms to uncover patterns in video game sales data. The optimal feature combination provided insights into regional sales correlations, and the evaluation metrics confirmed the robustness of the clusters formed. This approach provides a foundation for further exploration of sales trends and market segmentation in the gaming industry.

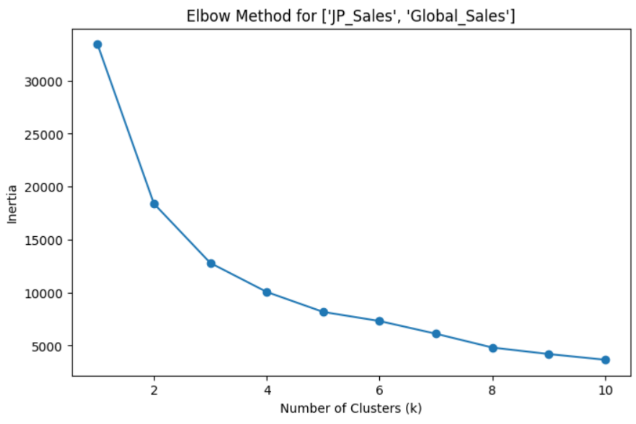
```
Clustering with features: ['NA_Sales', 'EU_Sales']
```



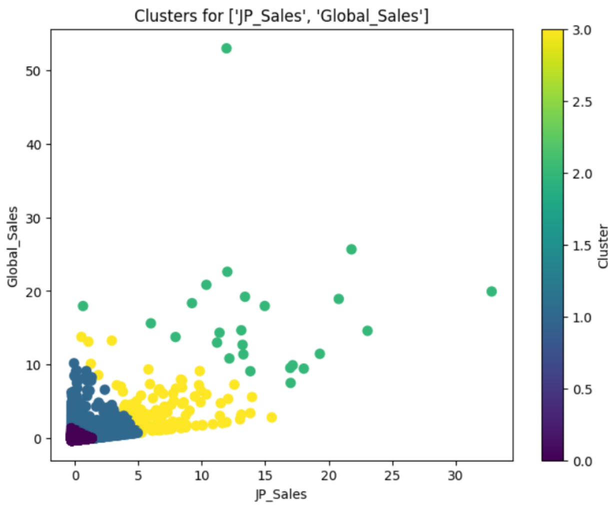
```
Optimal k: 4
Silhouette Score: 0.8332016110652556
Davies-Bouldin Index: 0.5598723310622952
```



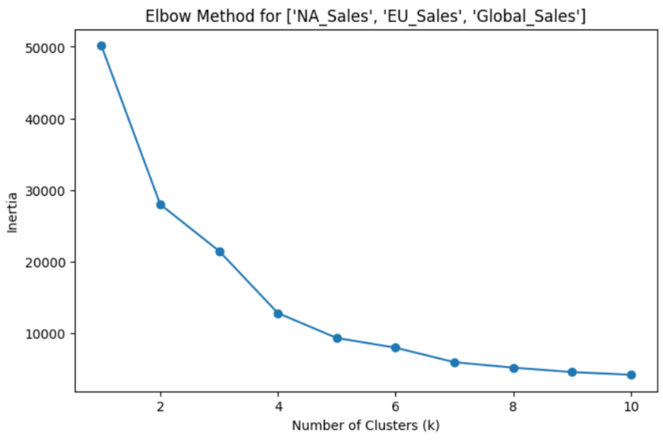
```
Clustering with features: ['JP_Sales', 'Global_Sales']
```



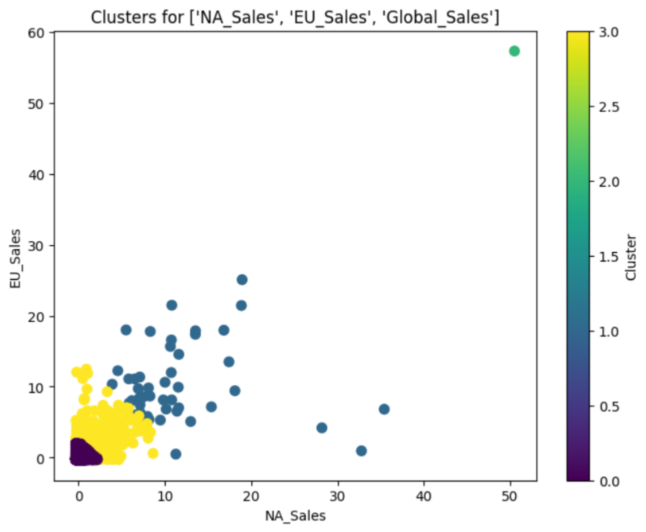
```
Optimal k: 4
Silhouette Score: 0.8042197504920231
Davies-Bouldin Index: 0.8488763228683637
```



```
Clustering with features: ['NA_Sales', 'EU_Sales', 'Global_Sales']
```



```
Optimal k: 4
Silhouette Score: 0.849001999727933
Davies-Bouldin Index: 0.502808379653066
```



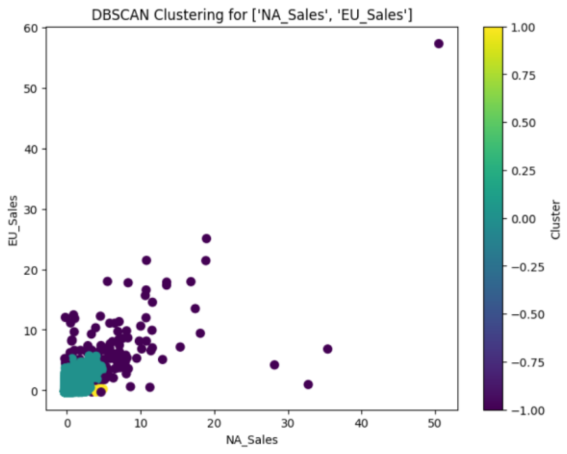
Comparing K-Means Clustering to Another Clustering Algorithm

In this part of the analysis, K-Means and DBSCAN clustering algorithms were applied to evaluate and compare their performance across selected feature combinations: ['NA_Sales', 'EU_Sales'], ['JP_Sales', 'Global_Sales'], and ['NA_Sales', 'EU_Sales', 'Global_Sales']. Both methods were assessed using the Silhouette Score, which evaluates cluster compactness and separation, and the Davies-Bouldin Index, which measures the quality of clustering with lower values indicating better separation.

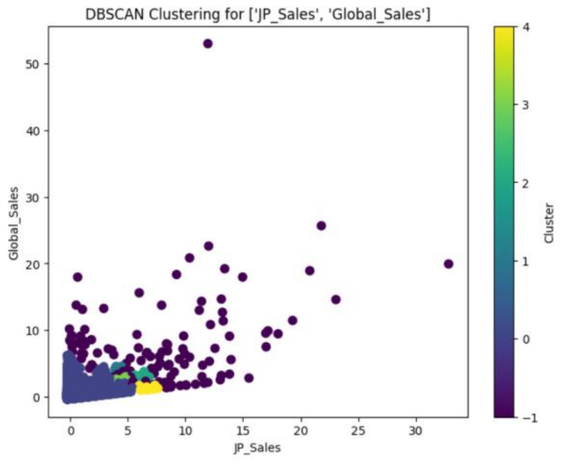
DBSCAN, a density-based algorithm, performed better for some feature combinations, particularly ['NA_Sales', 'EU_Sales'] (Silhouette Score = 0.858, Davies-Bouldin Index = 0.181) and ['JP_Sales', 'Global_Sales'] (Silhouette Score = 0.875, Davies-Bouldin Index = 0.376). DBSCAN effectively handled noise and identified denser clusters, outperforming K-Means in separation metrics for these combinations.

Overall, DBSCAN demonstrated superior performance in scenarios with distinct density variations, whereas K-Means maintained robust clustering with uniformly distributed data. The comparison highlights the importance of selecting an appropriate algorithm based on the dataset's characteristics, enabling better insights into video game sales trends and patterns.

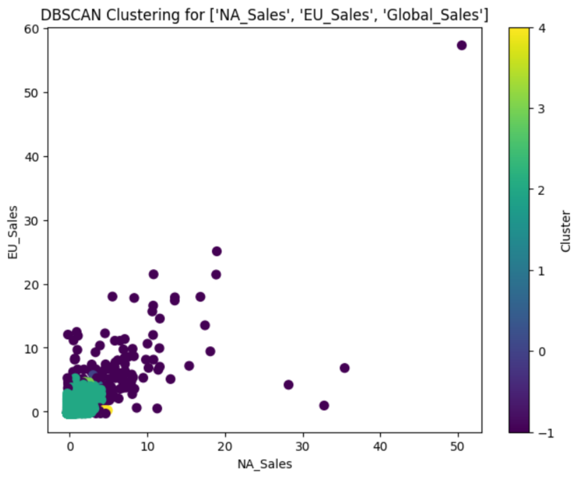
```
Clustering with features: ['NA_Sales', 'EU_Sales']
K-Means - Silhouette Score: 0.8332016110652556 , Davies-Bouldin Index: 0.5598723310622952
DBSCAN - Silhouette Score: 0.8581765321258263 , Davies-Bouldin Index: 0.1808189147294453
```

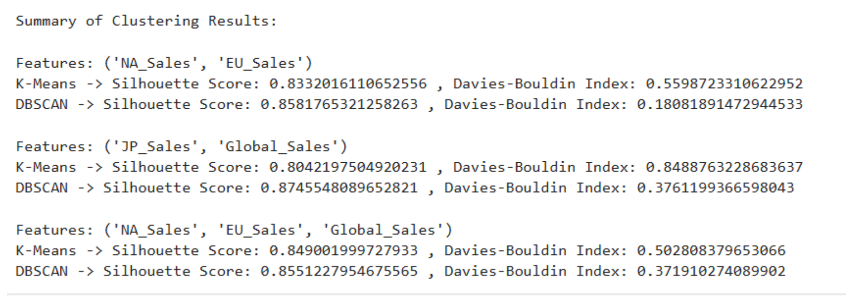


```
Clustering with features: ['JP_Sales', 'Global_Sales']
K-Means - Silhouette Score: 0.8042197504920231 , Davies-Bouldin Index: 0.8488763228683637
DBSCAN - Silhouette Score: 0.8745548089652821 , Davies-Bouldin Index: 0.3761199366598043
```



```
Clustering with features: ['NA_Sales', 'EU_Sales', 'Global_Sales']
K-Means - Silhouette Score: 0.849001999727933 , Davies-Bouldin Index: 0.502808379653066
DBSCAN - Silhouette Score: 0.8551227954675565 , Davies-Bouldin Index: 0.371910274089902
```





```
Summary of Clustering Results:

Features: ('NA_Sales', 'EU_Sales')
K-Means -> Silhouette Score: 0.8332016110652556 , Davies-Bouldin Index: 0.5598723310622952
DBSCAN -> Silhouette Score: 0.8581765321258263 , Davies-Bouldin Index: 0.18081891472944533

Features: ('JP_Sales', 'Global_Sales')
K-Means -> Silhouette Score: 0.8042197504920231 , Davies-Bouldin Index: 0.8488763228683637
DBSCAN -> Silhouette Score: 0.8745548089652821 , Davies-Bouldin Index: 0.3761199366598043

Features: ('NA_Sales', 'EU_Sales', 'Global_Sales')
K-Means -> Silhouette Score: 0.849001999727933 , Davies-Bouldin Index: 0.502808379653066
DBSCAN -> Silhouette Score: 0.8551227954675565 , Davies-Bouldin Index: 0.371910274089902
```

COMPONENT TWO (Emergency Vehicle Identification)

Architecture of CNN Model

In this task, Convolutional Neural Network (CNN) architecture aimed at classifying images of vehicles into emergency and non-emergency categories was designed and implemented. The CNN is built using TensorFlow and Keras and leverages essential deep learning components to achieve effective feature extraction and classification.

The CNN architecture consists of three convolutional layers, each followed by a max-pooling layer, a dense layer, and a dropout layer. The first convolutional layer uses 32 filters with a kernel size of 3x3, followed by a ReLU activation function to introduce non-linearity. Max-pooling layers are incorporated after each convolutional block to downsample feature maps, reducing spatial dimensions while preserving critical features. The number of filters increases progressively (32 → 64 → 128) as we move deeper into the network, allowing the model to capture more complex patterns.

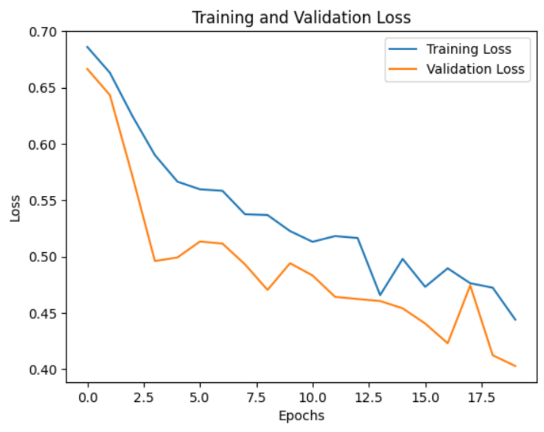
To prevent overfitting, a dropout layer with a 50% rate is included after the dense layer. A final dense layer with a single neuron and a sigmoid activation function outputs probabilities, making the model suitable for binary classification. The total parameter count for the model is 5.6 million, indicating a balance between model complexity and computational efficiency.

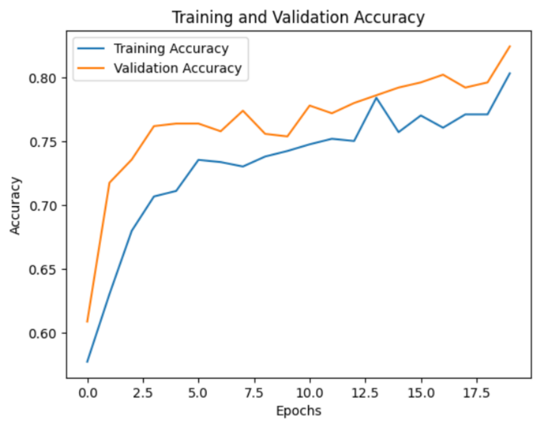
The model is compiled using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01 and momentum of 0.9. Binary cross-entropy is used as the loss function, given the binary nature of the classification task. The model is trained for 20 epochs using data augmentation techniques like rotation, width/height shifting, and zooming, which improve generalization by simulating variations in the training data.

Performance metrics, including training and validation accuracy and loss, were plotted. The plots reveal consistent improvement in performance over the epochs, with validation accuracy reaching approximately 80%. This indicates effective learning while avoiding overfitting.

Overall, the architecture and training approach provide a solid foundation for vehicle classification tasks, showcasing the efficacy of CNNs in feature extraction and decision-making for binary classification problems.

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 222, 222, 32)        │             896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 111, 111, 32)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 109, 109, 64)        │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 54, 54, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 52, 52, 128)         │          73,856 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 26, 26, 128)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 86528)               │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 64)                  │       5,537,856 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout (Dropout)                    │ (None, 64)                  │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 1)                   │              65 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘

 Total params: 5,631,169 (21.48 MB)
 Trainable params: 5,631,169 (21.48 MB)
 Non-trainable params: 0 (0.00 B)
36/36 ━━━━━━━━━━━━━━━━━━━━ 33s 785ms/step - accuracy: 0.5973 - loss: 0.6832 - val_accuracy: 0.6089 - val_loss: 0.6665
Epoch 2/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 32s 819ms/step - accuracy: 0.6243 - loss: 0.6686 - val_accuracy: 0.7177 - val_loss: 0.6434
Epoch 3/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 34s 864ms/step - accuracy: 0.6912 - loss: 0.6253 - val_accuracy: 0.7359 - val_loss: 0.5716
Epoch 4/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 32s 829ms/step - accuracy: 0.7066 - loss: 0.5878 - val_accuracy: 0.7621 - val_loss: 0.4961
Epoch 5/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 33s 838ms/step - accuracy: 0.6957 - loss: 0.5724 - val_accuracy: 0.7641 - val_loss: 0.4993
Epoch 6/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 37s 942ms/step - accuracy: 0.7492 - loss: 0.5753 - val_accuracy: 0.7641 - val_loss: 0.5134
Epoch 7/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 36s 931ms/step - accuracy: 0.7342 - loss: 0.5631 - val_accuracy: 0.7581 - val_loss: 0.5116
Epoch 8/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 36s 913ms/step - accuracy: 0.7357 - loss: 0.5249 - val_accuracy: 0.7742 - val_loss: 0.4931
Epoch 9/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 35s 910ms/step - accuracy: 0.7324 - loss: 0.5453 - val_accuracy: 0.7560 - val_loss: 0.4705
Epoch 10/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 37s 963ms/step - accuracy: 0.7433 - loss: 0.5237 - val_accuracy: 0.7540 - val_loss: 0.4941
Epoch 11/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 36s 933ms/step - accuracy: 0.7350 - loss: 0.5139 - val_accuracy: 0.7782 - val_loss: 0.4833
Epoch 12/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 35s 889ms/step - accuracy: 0.7732 - loss: 0.4825 - val_accuracy: 0.7722 - val_loss: 0.4643
Epoch 13/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.7569 - loss: 0.5048 - val_accuracy: 0.7802 - val_loss: 0.4624
Epoch 14/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 38s 994ms/step - accuracy: 0.7619 - loss: 0.4847 - val_accuracy: 0.7863 - val_loss: 0.4606
Epoch 15/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 46s 1s/step - accuracy: 0.7556 - loss: 0.4949 - val_accuracy: 0.7923 - val_loss: 0.4541
Epoch 16/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 43s 1s/step - accuracy: 0.7761 - loss: 0.4776 - val_accuracy: 0.7964 - val_loss: 0.4406
Epoch 17/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.7738 - loss: 0.4847 - val_accuracy: 0.8024 - val_loss: 0.4231
Epoch 18/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 36s 920ms/step - accuracy: 0.7865 - loss: 0.4533 - val_accuracy: 0.7923 - val_loss: 0.4744
Epoch 19/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 36s 919ms/step - accuracy: 0.7557 - loss: 0.4910 - val_accuracy: 0.7964 - val_loss: 0.4125
Epoch 20/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 34s 872ms/step - accuracy: 0.8143 - loss: 0.4476 - val_accuracy: 0.8246 - val_loss: 0.4029
```





Regularisation Method

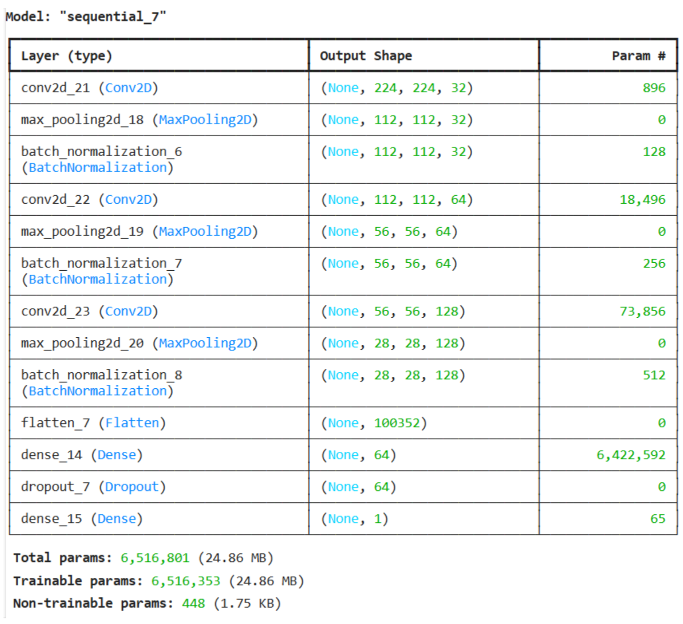
Regularization techniques were pivotal in improving model performance and reducing overfitting. Two major methods were employed: dropout and batch normalization.

1. Dropout: A dropout rate of 0.5 was used to randomly deactivate neurons during training. This introduced noise and forced the network to rely on diverse neurons for feature extraction, enhancing generalization. As a result, training and validation accuracy curves demonstrated reduced overfitting, with loss metrics stabilizing across epochs.

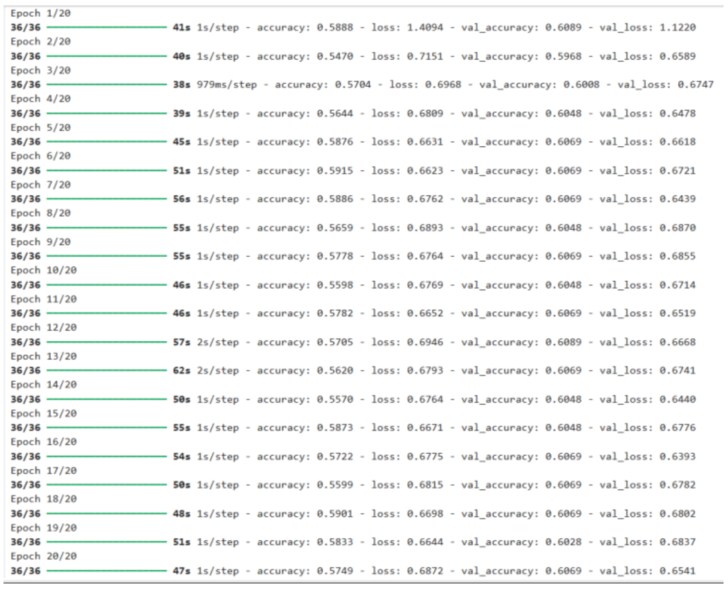
2. Batch Normalization: Batch normalization layers normalized intermediate feature maps, addressing internal covariate shifts. This technique enhanced training stability and allowed higher learning rates. The validation loss remained low, and the model achieved consistent accuracy improvements over epochs.

Together, these methods substantially improved the CNN's robustness, resulting in reliable predictions on unseen data, as reflected in the accuracy and loss plots.

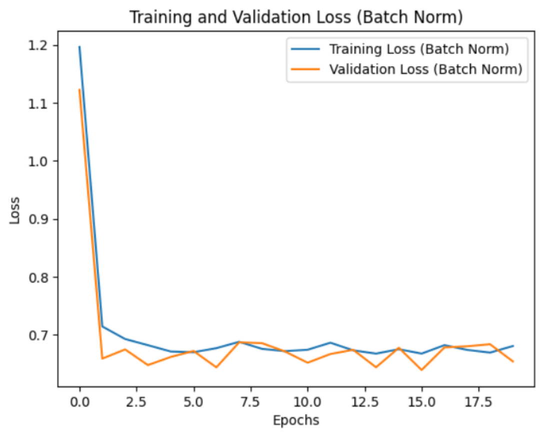
Batch Normalisation for Regularisation

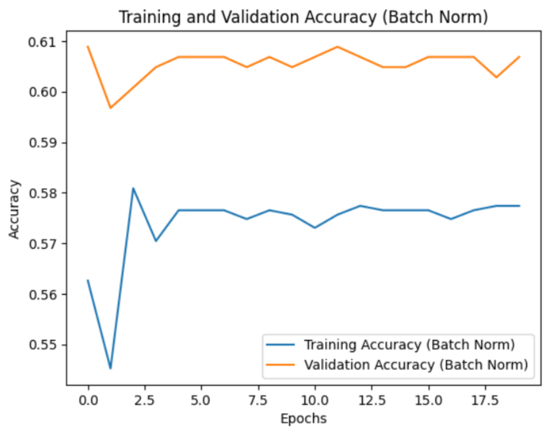


```
Model: "sequential_7"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_21 (Conv2D)                   │ (None, 224, 224, 32)        │             896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_18 (MaxPooling2D)      │ (None, 112, 112, 32)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_6                │ (None, 112, 112, 32)        │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_22 (Conv2D)                   │ (None, 112, 112, 64)        │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_19 (MaxPooling2D)      │ (None, 56, 56, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_7                │ (None, 56, 56, 64)          │             256 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_23 (Conv2D)                   │ (None, 56, 56, 128)         │          73,856 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_20 (MaxPooling2D)      │ (None, 28, 28, 128)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_8                │ (None, 28, 28, 128)         │             512 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten_7 (Flatten)                  │ (None, 100352)              │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_14 (Dense)                     │ (None, 64)                  │       6,422,592 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_7 (Dropout)                  │ (None, 64)                  │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_15 (Dense)                     │ (None, 1)                   │              65 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 6,516,801 (24.86 MB)
 Trainable params: 6,516,353 (24.86 MB)
 Non-trainable params: 448 (1.75 KB)
```



```
Epoch 1/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.5888 - loss: 1.4094 - val_accuracy: 0.6089 - val_loss: 1.1220
Epoch 2/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.5470 - loss: 0.7151 - val_accuracy: 0.5968 - val_loss: 0.6589
Epoch 3/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 38s 979ms/step - accuracy: 0.5704 - loss: 0.6968 - val_accuracy: 0.6008 - val_loss: 0.6747
Epoch 4/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.5644 - loss: 0.6809 - val_accuracy: 0.6048 - val_loss: 0.6478
Epoch 5/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.5876 - loss: 0.6631 - val_accuracy: 0.6069 - val_loss: 0.6618
Epoch 6/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 51s 1s/step - accuracy: 0.5915 - loss: 0.6623 - val_accuracy: 0.6069 - val_loss: 0.6721
Epoch 7/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 56s 1s/step - accuracy: 0.5886 - loss: 0.6762 - val_accuracy: 0.6069 - val_loss: 0.6439
Epoch 8/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 55s 1s/step - accuracy: 0.5659 - loss: 0.6893 - val_accuracy: 0.6048 - val_loss: 0.6870
Epoch 9/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 55s 1s/step - accuracy: 0.5778 - loss: 0.6764 - val_accuracy: 0.6069 - val_loss: 0.6855
Epoch 10/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 46s 1s/step - accuracy: 0.5598 - loss: 0.6769 - val_accuracy: 0.6048 - val_loss: 0.6714
Epoch 11/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 46s 1s/step - accuracy: 0.5782 - loss: 0.6652 - val_accuracy: 0.6069 - val_loss: 0.6519
Epoch 12/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 57s 2s/step - accuracy: 0.5705 - loss: 0.6946 - val_accuracy: 0.6089 - val_loss: 0.6668
Epoch 13/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 62s 2s/step - accuracy: 0.5620 - loss: 0.6793 - val_accuracy: 0.6069 - val_loss: 0.6741
Epoch 14/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 50s 1s/step - accuracy: 0.5570 - loss: 0.6764 - val_accuracy: 0.6048 - val_loss: 0.6440
Epoch 15/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 55s 1s/step - accuracy: 0.5873 - loss: 0.6671 - val_accuracy: 0.6048 - val_loss: 0.6776
Epoch 16/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 54s 1s/step - accuracy: 0.5722 - loss: 0.6775 - val_accuracy: 0.6069 - val_loss: 0.6393
Epoch 17/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 50s 1s/step - accuracy: 0.5599 - loss: 0.6815 - val_accuracy: 0.6069 - val_loss: 0.6782
Epoch 18/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 48s 1s/step - accuracy: 0.5901 - loss: 0.6698 - val_accuracy: 0.6069 - val_loss: 0.6802
Epoch 19/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 51s 1s/step - accuracy: 0.5833 - loss: 0.6644 - val_accuracy: 0.6028 - val_loss: 0.6837
Epoch 20/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 47s 1s/step - accuracy: 0.5749 - loss: 0.6872 - val_accuracy: 0.6069 - val_loss: 0.6541
```





Hyperparameter Tuning

To enhance the performance of the CNN model, several hyperparameter tuning strategies were applied, focusing on parameters like batch size, kernel size, stride, and pooling techniques. Each adjustment was systematically tested to determine its influence on the model's accuracy and loss.

1. **Batch Size Optimization:** Initially, a batch size of 128 was tested. This choice aimed to balance computational efficiency and model generalization. The results showed improved validation accuracy and reduced overfitting compared to smaller batch sizes. Accuracy trends indicated steady improvement over epochs, with both training and validation accuracies aligning closely, suggesting effective optimization.

2. **Kernel Size Variation:** The kernel size was expanded to 5X5 to analyse its effect on feature extraction. Larger kernels capture broader spatial features, which can be beneficial for certain datasets. The model with a 5X5 kernel size achieved improved training and validation accuracies compared to the default 3X3 kernel size. However, this change also led to an increase in model complexity, as reflected by the higher parameter count.

3. **Stride Adjustment:** Stride was adjusted to 2X2 while maintaining the kernel size at 3X3. This increased the down-sampling effect, reducing computational load and model size.

Although the validation accuracy was slightly lower than with the original stride of 1X1, the model demonstrated faster convergence and reduced training time. This trade-off was considered valuable in resource-constrained environments.

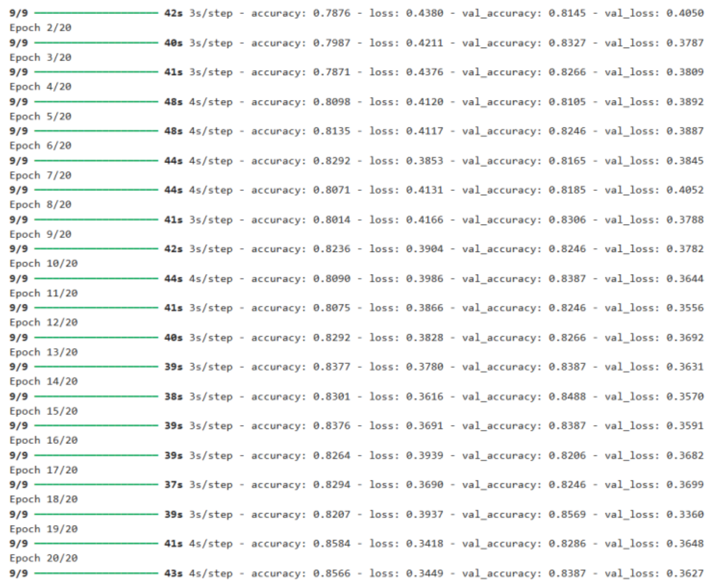
4. **Pooling Layer Modification:** An alternative pooling technique, average pooling, was tested against max pooling. While max pooling emphasizes prominent features, average pooling provides a more balanced feature representation. The model with average pooling exhibited stable training but slightly lower accuracy compared to max pooling, indicating that max pooling is better suited for this dataset.

Figures and Performance Visualization:

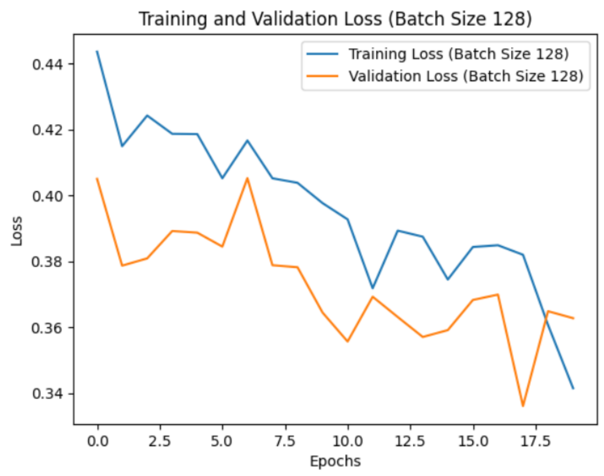
- Training and validation accuracy curves for each configuration highlighted the effects of hyperparameter tuning. The final model, optimized with max pooling, 5X5 kernels, and a stride of 1X1, achieved the highest validation accuracy, with minimal overfitting.
- Loss curves further emphasized the effectiveness of tuning in reducing both training and validation losses, reflecting enhanced model stability and performance.

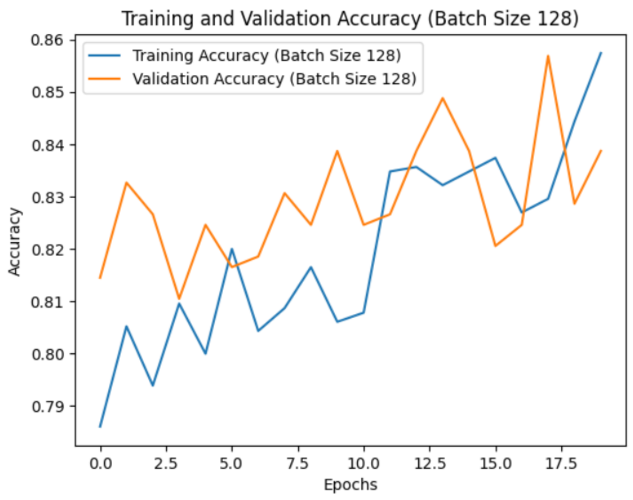
In conclusion, kernel size and pooling strategy exhibited the strongest influence on model performance, followed by batch size and stride adjustments. These findings emphasize the importance of systematic experimentation for optimal model configuration.

Batch size Initialised to 128

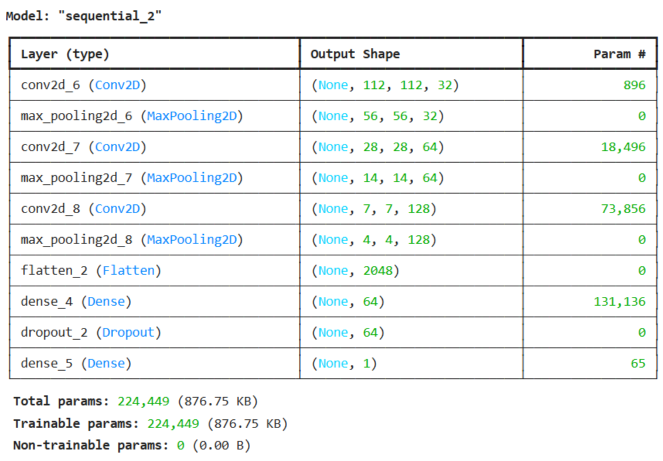


```
9/9 ━━━━━━━━━━━━━━━━━━━━ 42s 3s/step - accuracy: 0.7876 - loss: 0.4380 - val_accuracy: 0.8145 - val_loss: 0.4050
Epoch 2/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 40s 3s/step - accuracy: 0.7987 - loss: 0.4211 - val_accuracy: 0.8327 - val_loss: 0.3787
Epoch 3/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 41s 3s/step - accuracy: 0.7871 - loss: 0.4376 - val_accuracy: 0.8266 - val_loss: 0.3809
Epoch 4/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 48s 4s/step - accuracy: 0.8098 - loss: 0.4120 - val_accuracy: 0.8105 - val_loss: 0.3892
Epoch 5/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 48s 4s/step - accuracy: 0.8135 - loss: 0.4117 - val_accuracy: 0.8246 - val_loss: 0.3887
Epoch 6/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 44s 4s/step - accuracy: 0.8292 - loss: 0.3853 - val_accuracy: 0.8165 - val_loss: 0.3845
Epoch 7/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 44s 4s/step - accuracy: 0.8071 - loss: 0.4131 - val_accuracy: 0.8185 - val_loss: 0.4052
Epoch 8/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 41s 3s/step - accuracy: 0.8014 - loss: 0.4166 - val_accuracy: 0.8306 - val_loss: 0.3788
Epoch 9/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 42s 3s/step - accuracy: 0.8236 - loss: 0.3904 - val_accuracy: 0.8246 - val_loss: 0.3782
Epoch 10/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 44s 4s/step - accuracy: 0.8090 - loss: 0.3986 - val_accuracy: 0.8387 - val_loss: 0.3644
Epoch 11/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 41s 3s/step - accuracy: 0.8075 - loss: 0.3866 - val_accuracy: 0.8246 - val_loss: 0.3556
Epoch 12/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 40s 3s/step - accuracy: 0.8292 - loss: 0.3828 - val_accuracy: 0.8266 - val_loss: 0.3692
Epoch 13/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 39s 3s/step - accuracy: 0.8377 - loss: 0.3780 - val_accuracy: 0.8387 - val_loss: 0.3631
Epoch 14/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 38s 3s/step - accuracy: 0.8301 - loss: 0.3616 - val_accuracy: 0.8488 - val_loss: 0.3570
Epoch 15/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 39s 3s/step - accuracy: 0.8376 - loss: 0.3691 - val_accuracy: 0.8387 - val_loss: 0.3591
Epoch 16/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 39s 3s/step - accuracy: 0.8264 - loss: 0.3939 - val_accuracy: 0.8206 - val_loss: 0.3682
Epoch 17/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 37s 3s/step - accuracy: 0.8294 - loss: 0.3690 - val_accuracy: 0.8246 - val_loss: 0.3699
Epoch 18/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 39s 3s/step - accuracy: 0.8207 - loss: 0.3937 - val_accuracy: 0.8569 - val_loss: 0.3360
Epoch 19/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 41s 4s/step - accuracy: 0.8584 - loss: 0.3418 - val_accuracy: 0.8286 - val_loss: 0.3648
Epoch 20/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 43s 4s/step - accuracy: 0.8566 - loss: 0.3449 - val_accuracy: 0.8387 - val_loss: 0.3627
```

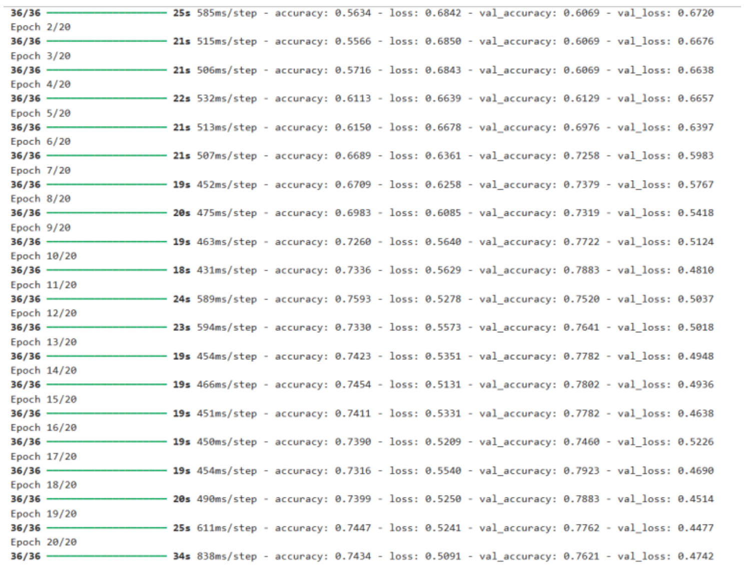




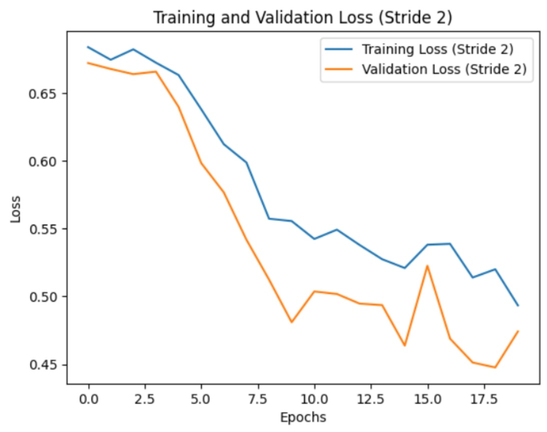
Rebuilding the CNN model with a stride length of 2 and padding

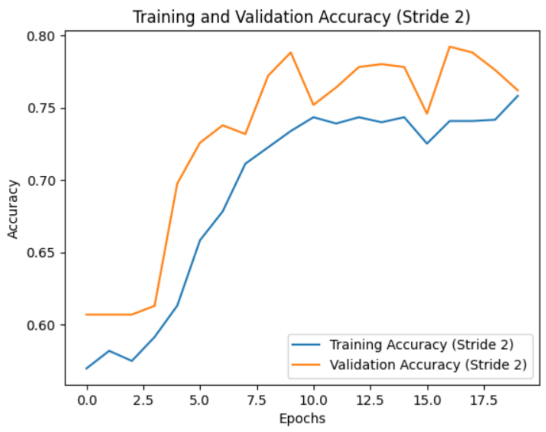


```
Model: "sequential_2"

| Layer (type)                     | Output Shape                | Param # |
|----------------------------------|-----------------------------|---------|
| conv2d_6 (Conv2D)                | (None, 112, 112, 32)        |     896 |
| max_pooling2d_6 (MaxPooling2D)   | (None, 56, 56, 32)          |       0 |
| conv2d_7 (Conv2D)                | (None, 28, 28, 64)          |  18,496 |
| max_pooling2d_7 (MaxPooling2D)   | (None, 14, 14, 64)          |       0 |
| conv2d_8 (Conv2D)                | (None, 7, 7, 128)           |  73,856 |
| max_pooling2d_8 (MaxPooling2D)   | (None, 4, 4, 128)           |       0 |
| flatten_2 (Flatten)              | (None, 2048)                |       0 |
| dense_4 (Dense)                  | (None, 64)                  | 131,136 |
| dropout_2 (Dropout)              | (None, 64)                  |       0 |
| dense_5 (Dense)                  | (None, 1)                   |      65 |

 Total params: 224,449 (876.75 KB)
 Trainable params: 224,449 (876.75 KB)
 Non-trainable params: 0 (0.00 B)
```

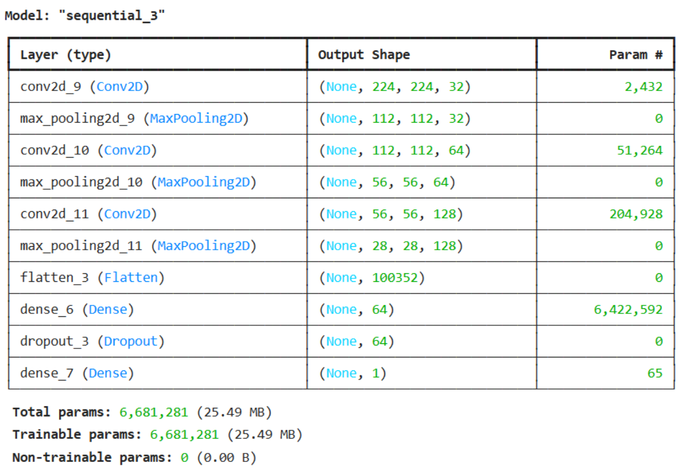


```
36/36 ━━━━━━━━━━━━━━━━━━━━ 25s 585ms/step - accuracy: 0.5634 - loss: 0.6842 - val_accuracy: 0.6069 - val_loss: 0.6720
Epoch 2/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 21s 515ms/step - accuracy: 0.5566 - loss: 0.6850 - val_accuracy: 0.6069 - val_loss: 0.6676
Epoch 3/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 21s 506ms/step - accuracy: 0.5716 - loss: 0.6843 - val_accuracy: 0.6069 - val_loss: 0.6638
Epoch 4/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 22s 532ms/step - accuracy: 0.6113 - loss: 0.6639 - val_accuracy: 0.6129 - val_loss: 0.6657
Epoch 5/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 21s 513ms/step - accuracy: 0.6150 - loss: 0.6678 - val_accuracy: 0.6976 - val_loss: 0.6397
Epoch 6/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 21s 507ms/step - accuracy: 0.6689 - loss: 0.6361 - val_accuracy: 0.7258 - val_loss: 0.5983
Epoch 7/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 19s 452ms/step - accuracy: 0.6709 - loss: 0.6258 - val_accuracy: 0.7379 - val_loss: 0.5767
Epoch 8/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 20s 475ms/step - accuracy: 0.6983 - loss: 0.6085 - val_accuracy: 0.7319 - val_loss: 0.5418
Epoch 9/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 19s 463ms/step - accuracy: 0.7260 - loss: 0.5640 - val_accuracy: 0.7722 - val_loss: 0.5124
Epoch 10/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 18s 431ms/step - accuracy: 0.7336 - loss: 0.5629 - val_accuracy: 0.7883 - val_loss: 0.4810
Epoch 11/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 24s 589ms/step - accuracy: 0.7593 - loss: 0.5278 - val_accuracy: 0.7520 - val_loss: 0.5037
Epoch 12/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 23s 594ms/step - accuracy: 0.7330 - loss: 0.5573 - val_accuracy: 0.7641 - val_loss: 0.5018
Epoch 13/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 19s 454ms/step - accuracy: 0.7423 - loss: 0.5351 - val_accuracy: 0.7782 - val_loss: 0.4948
Epoch 14/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 19s 466ms/step - accuracy: 0.7454 - loss: 0.5131 - val_accuracy: 0.7802 - val_loss: 0.4936
Epoch 15/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 19s 451ms/step - accuracy: 0.7411 - loss: 0.5331 - val_accuracy: 0.7782 - val_loss: 0.4638
Epoch 16/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 19s 450ms/step - accuracy: 0.7390 - loss: 0.5209 - val_accuracy: 0.7460 - val_loss: 0.5226
Epoch 17/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 19s 454ms/step - accuracy: 0.7316 - loss: 0.5540 - val_accuracy: 0.7923 - val_loss: 0.4690
Epoch 18/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 20s 490ms/step - accuracy: 0.7399 - loss: 0.5250 - val_accuracy: 0.7883 - val_loss: 0.4514
Epoch 19/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 25s 611ms/step - accuracy: 0.7447 - loss: 0.5241 - val_accuracy: 0.7762 - val_loss: 0.4477
Epoch 20/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 34s 838ms/step - accuracy: 0.7434 - loss: 0.5091 - val_accuracy: 0.7621 - val_loss: 0.4742
```

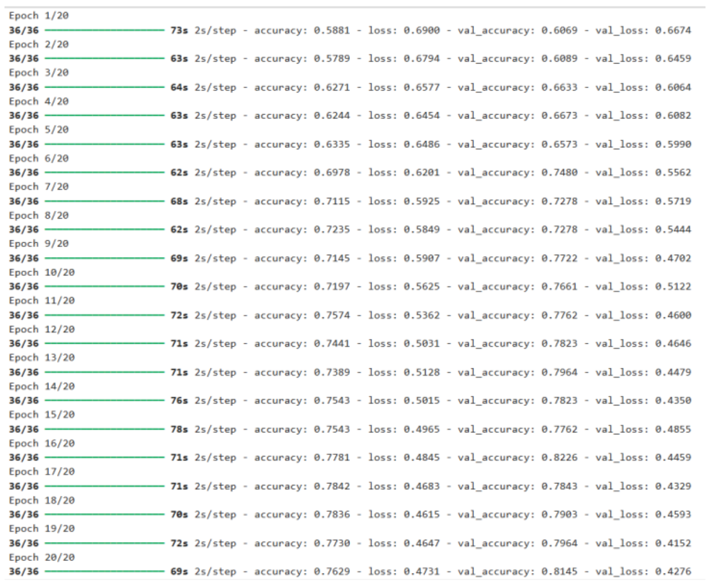




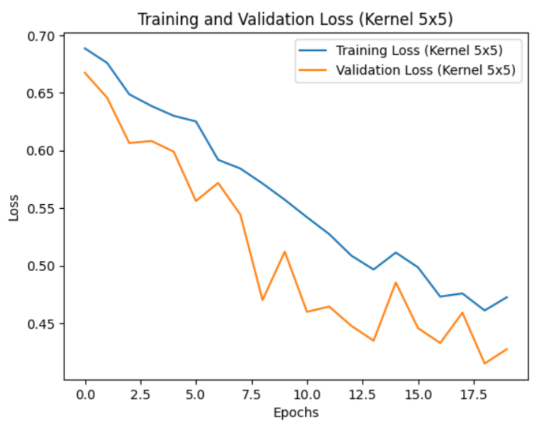
Rebuilding the CNN model with kernel size 5x5

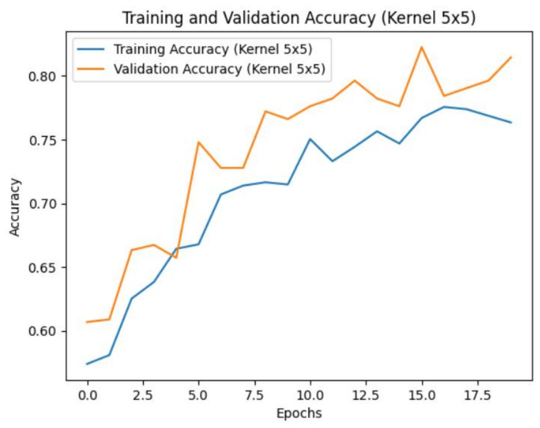


```
Model: "sequential_3"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                        ┃ Output Shape                 ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_9 (Conv2D)                   │ (None, 224, 224, 32)         │           2,432 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ max_pooling2d_9 (MaxPooling2D)      │ (None, 112, 112, 32)         │               0 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ conv2d_10 (Conv2D)                  │ (None, 112, 112, 64)         │          51,264 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ max_pooling2d_10 (MaxPooling2D)     │ (None, 56, 56, 64)           │               0 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ conv2d_11 (Conv2D)                  │ (None, 56, 56, 128)          │         204,928 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ max_pooling2d_11 (MaxPooling2D)     │ (None, 28, 28, 128)          │               0 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ flatten_3 (Flatten)                 │ (None, 100352)               │               0 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ dense_6 (Dense)                     │ (None, 64)                   │       6,422,592 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ dropout_3 (Dropout)                 │ (None, 64)                   │               0 │
├─────────────────────────────────────┼──────────────────────────────┼─────────────────┤
│ dense_7 (Dense)                     │ (None, 1)                    │              65 │
└─────────────────────────────────────┴──────────────────────────────┴─────────────────┘

 Total params: 6,681,281 (25.49 MB)
 Trainable params: 6,681,281 (25.49 MB)
 Non-trainable params: 0 (0.00 B)
```

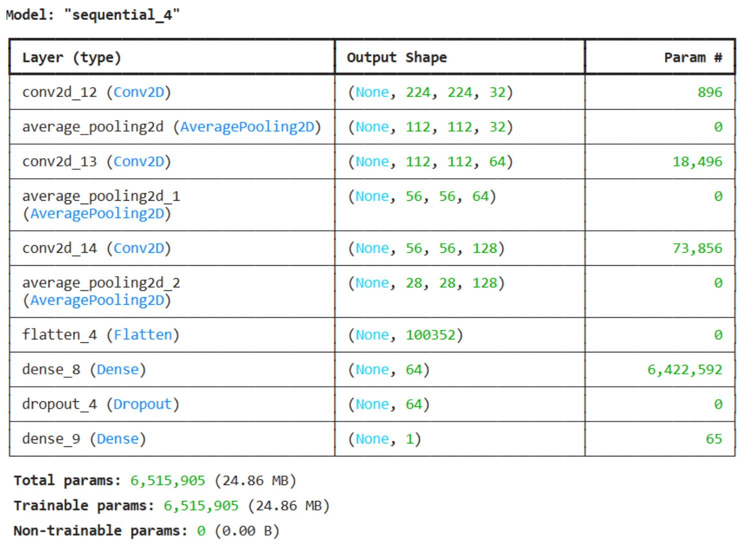


```
Epoch 1/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 73s 2s/step - accuracy: 0.5881 - loss: 0.6900 - val_accuracy: 0.6069 - val_loss: 0.6674
Epoch 2/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 63s 2s/step - accuracy: 0.5789 - loss: 0.6794 - val_accuracy: 0.6089 - val_loss: 0.6459
Epoch 3/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 64s 2s/step - accuracy: 0.6271 - loss: 0.6577 - val_accuracy: 0.6633 - val_loss: 0.6064
Epoch 4/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 63s 2s/step - accuracy: 0.6244 - loss: 0.6454 - val_accuracy: 0.6673 - val_loss: 0.6082
Epoch 5/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 63s 2s/step - accuracy: 0.6335 - loss: 0.6486 - val_accuracy: 0.6573 - val_loss: 0.5990
Epoch 6/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 62s 2s/step - accuracy: 0.6978 - loss: 0.6201 - val_accuracy: 0.7480 - val_loss: 0.5562
Epoch 7/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 68s 2s/step - accuracy: 0.7115 - loss: 0.5925 - val_accuracy: 0.7278 - val_loss: 0.5719
Epoch 8/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 62s 2s/step - accuracy: 0.7235 - loss: 0.5849 - val_accuracy: 0.7278 - val_loss: 0.5444
Epoch 9/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 69s 2s/step - accuracy: 0.7145 - loss: 0.5907 - val_accuracy: 0.7722 - val_loss: 0.4702
Epoch 10/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 70s 2s/step - accuracy: 0.7197 - loss: 0.5625 - val_accuracy: 0.7661 - val_loss: 0.5122
Epoch 11/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 72s 2s/step - accuracy: 0.7574 - loss: 0.5362 - val_accuracy: 0.7762 - val_loss: 0.4600
Epoch 12/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 71s 2s/step - accuracy: 0.7441 - loss: 0.5031 - val_accuracy: 0.7823 - val_loss: 0.4646
Epoch 13/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 71s 2s/step - accuracy: 0.7389 - loss: 0.5128 - val_accuracy: 0.7964 - val_loss: 0.4479
Epoch 14/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 76s 2s/step - accuracy: 0.7543 - loss: 0.5015 - val_accuracy: 0.7823 - val_loss: 0.4350
Epoch 15/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 78s 2s/step - accuracy: 0.7543 - loss: 0.4965 - val_accuracy: 0.7762 - val_loss: 0.4855
Epoch 16/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 71s 2s/step - accuracy: 0.7781 - loss: 0.4845 - val_accuracy: 0.8226 - val_loss: 0.4459
Epoch 17/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 71s 2s/step - accuracy: 0.7842 - loss: 0.4683 - val_accuracy: 0.7843 - val_loss: 0.4329
Epoch 18/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 70s 2s/step - accuracy: 0.7836 - loss: 0.4615 - val_accuracy: 0.7903 - val_loss: 0.4593
Epoch 19/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 72s 2s/step - accuracy: 0.7730 - loss: 0.4647 - val_accuracy: 0.7964 - val_loss: 0.4152
Epoch 20/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 69s 2s/step - accuracy: 0.7629 - loss: 0.4731 - val_accuracy: 0.8145 - val_loss: 0.4276
```





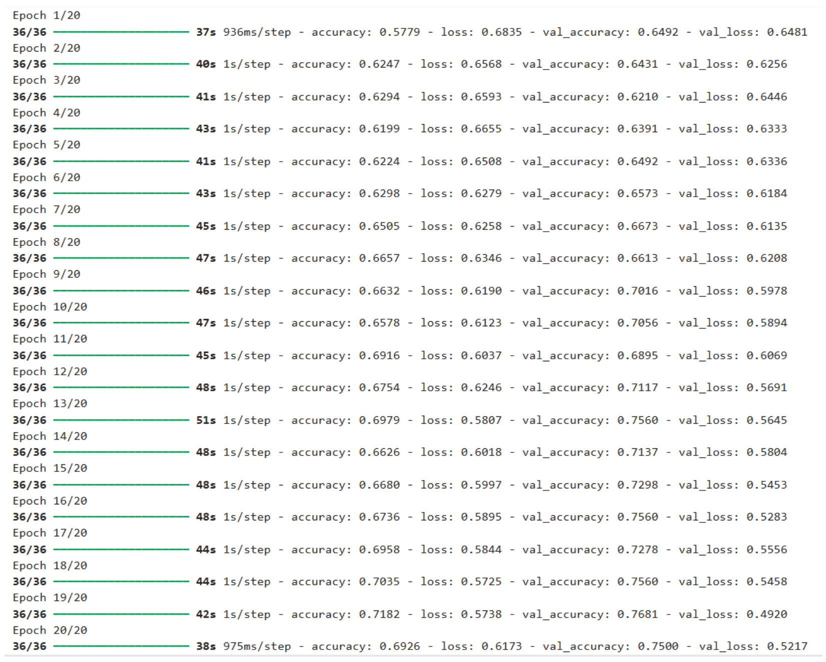
Rebuilding the CNN model with Average Pooling



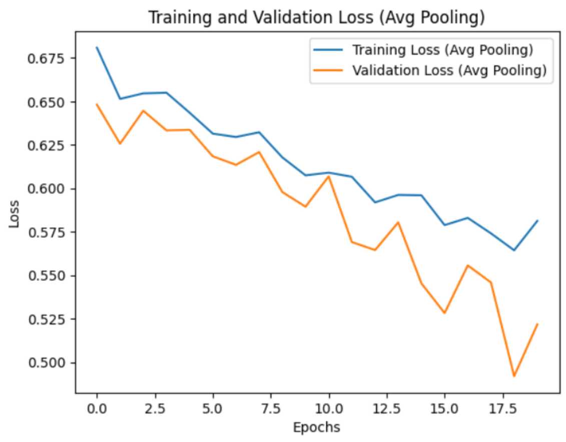
Model: "sequential_4"

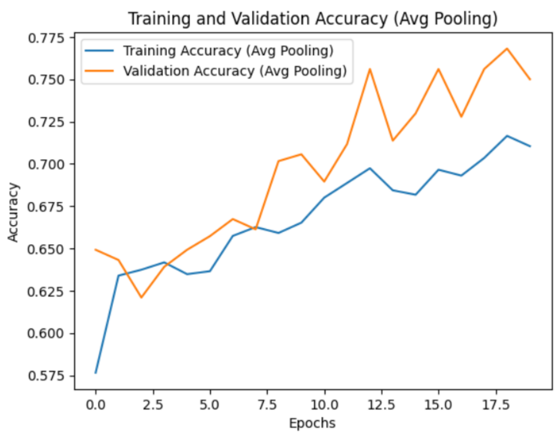
Layer (type)	**Output Shape**	**Param #**
conv2d_12 (Conv2D)	(None, 224, 224, 32)	896
average_pooling2d (AveragePooling2D)	(None, 112, 112, 32)	0
conv2d_13 (Conv2D)	(None, 112, 112, 64)	18,496
average_pooling2d_1 (AveragePooling2D)	(None, 56, 56, 64)	0
conv2d_14 (Conv2D)	(None, 56, 56, 128)	73,856
average_pooling2d_2 (AveragePooling2D)	(None, 28, 28, 128)	0
flatten_4 (Flatten)	(None, 100352)	0
dense_8 (Dense)	(None, 64)	6,422,592
dropout_4 (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 1)	65

Total params: 6,515,905 (24.86 MB)
Trainable params: 6,515,905 (24.86 MB)
Non-trainable params: 0 (0.00 B)



```
Epoch 1/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 37s 936ms/step - accuracy: 0.5779 - loss: 0.6835 - val_accuracy: 0.6492 - val_loss: 0.6481
Epoch 2/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.6247 - loss: 0.6568 - val_accuracy: 0.6431 - val_loss: 0.6256
Epoch 3/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.6294 - loss: 0.6593 - val_accuracy: 0.6210 - val_loss: 0.6446
Epoch 4/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 43s 1s/step - accuracy: 0.6199 - loss: 0.6655 - val_accuracy: 0.6391 - val_loss: 0.6333
Epoch 5/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.6224 - loss: 0.6508 - val_accuracy: 0.6492 - val_loss: 0.6336
Epoch 6/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 43s 1s/step - accuracy: 0.6298 - loss: 0.6279 - val_accuracy: 0.6573 - val_loss: 0.6184
Epoch 7/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.6505 - loss: 0.6258 - val_accuracy: 0.6673 - val_loss: 0.6135
Epoch 8/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 47s 1s/step - accuracy: 0.6657 - loss: 0.6346 - val_accuracy: 0.6613 - val_loss: 0.6208
Epoch 9/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 46s 1s/step - accuracy: 0.6632 - loss: 0.6190 - val_accuracy: 0.7016 - val_loss: 0.5978
Epoch 10/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 47s 1s/step - accuracy: 0.6578 - loss: 0.6123 - val_accuracy: 0.7056 - val_loss: 0.5894
Epoch 11/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.6916 - loss: 0.6037 - val_accuracy: 0.6895 - val_loss: 0.6069
Epoch 12/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 48s 1s/step - accuracy: 0.6754 - loss: 0.6246 - val_accuracy: 0.7117 - val_loss: 0.5691
Epoch 13/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 51s 1s/step - accuracy: 0.6979 - loss: 0.5807 - val_accuracy: 0.7560 - val_loss: 0.5645
Epoch 14/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 48s 1s/step - accuracy: 0.6626 - loss: 0.6018 - val_accuracy: 0.7137 - val_loss: 0.5804
Epoch 15/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 48s 1s/step - accuracy: 0.6680 - loss: 0.5997 - val_accuracy: 0.7298 - val_loss: 0.5453
Epoch 16/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 48s 1s/step - accuracy: 0.6736 - loss: 0.5895 - val_accuracy: 0.7560 - val_loss: 0.5283
Epoch 17/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 44s 1s/step - accuracy: 0.6958 - loss: 0.5844 - val_accuracy: 0.7278 - val_loss: 0.5556
Epoch 18/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 44s 1s/step - accuracy: 0.7035 - loss: 0.5725 - val_accuracy: 0.7560 - val_loss: 0.5458
Epoch 19/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 42s 1s/step - accuracy: 0.7182 - loss: 0.5738 - val_accuracy: 0.7681 - val_loss: 0.4920
Epoch 20/20
36/36 ━━━━━━━━━━━━━━━━━━━━ 38s 975ms/step - accuracy: 0.6926 - loss: 0.6173 - val_accuracy: 0.7500 - val_loss: 0.5217
```





Evidence of Overfitting

COMPONENT THREE (Ethics of AI)

Technical Challenges

The integration of Explainable AI (XAI) in healthcare poses significant technical challenges, especially as it strives to balance advanced performance with the transparency needed for critical decision-making. Explainable AI ensures that AI systems provide clear and understandable reasoning behind their predictions or decisions, a necessity in an industry where trust and accountability are paramount (Holzinger et al., 2017).

One major complexity lies in the nature of deep learning models, such as neural networks, which are often considered "black boxes." These models excel in tasks like medical image analysis or disease prediction due to their ability to learn complex patterns in data. However, their intricate architectures make it challenging to articulate how they arrive at specific conclusions (Samek et al., 2017). This lack of interpretability undermines clinicians' trust and makes regulatory approval difficult.

Another challenge involves the trade-off between accuracy and interpretability. Simplifying models to enhance explainability often reduces their predictive power. For example, linear regression models are interpretable but may not perform as well as deep learning models in diagnosing diseases from imaging data. Striking a balance is critical yet technically demanding (Arrieta et al., 2020).

Moreover, healthcare data is characterized by its high dimensionality and heterogeneity, encompassing structured data (e.g., electronic health records) and unstructured data (e.g., imaging, text). Generating meaningful explanations from such diverse datasets requires advanced XAI techniques, such as attention mechanisms or feature attribution methods, which are computationally intensive (Tjoa and Guan, 2020).

Finally, integrating XAI into clinical workflows introduces challenges in standardization and usability. Explanations must align with the cognitive needs of healthcare professionals and comply with strict ethical and legal standards, such as the General Data Protection Regulation (GDPR). Developing universally accepted frameworks for generating and presenting explanations remains an ongoing technical hurdle (Topol, 2019).

Ethical Framework Evaluation

Ethical frameworks provide structured guidelines to ensure responsible AI development and deployment in healthcare. While they address critical ethical concerns like fairness, transparency, and accountability, they also face significant limitations in practical implementation.

The principle-based frameworks, such as the ones proposed by the European Commission’s Ethics Guidelines for Trustworthy AI, are notable for their clarity and broad applicability. These frameworks emphasize fairness, privacy, and transparency, which are essential in a sensitive industry like healthcare (European Commission, 2019). However, a limitation is their lack of specificity for complex healthcare contexts. For example, while they highlight the need for fairness, they do not sufficiently address biases in training data, which can perpetuate inequities in AI systems (Floridi et al., 2018).

Another framework, the General Data Protection Regulation (GDPR), ensures data privacy and security, pivotal in handling sensitive patient data. However, its strict regulations can hinder innovation, as developers may struggle to balance compliance with the iterative needs of AI development (Topol, 2019). Moreover, GDPR primarily focuses on data usage but provides limited guidance on mitigating biases and ensuring model fairness.

The Explainable AI (XAI) frameworks emphasize interpretability to foster trust among clinicians and patients. While beneficial in increasing system transparency, these frameworks often fail to achieve scalability in real-world applications. Advanced healthcare models like deep neural networks are inherently complex, making detailed explanations difficult without compromising performance (Holzinger et al., 2017).

In summary, these frameworks establish foundational ethical principles but often lack actionable, context-specific solutions for real-world healthcare challenges. Addressing these gaps requires iterative frameworks that incorporate domain-specific nuances and interdisciplinary collaboration between AI developers, clinicians, and policymakers.

Innovation and Solutions

Addressing the limitations in existing ethical frameworks for AI in healthcare requires targeted innovations that ensure practical, scalable, and context-specific solutions.

One potential innovation is the development of adaptive ethical frameworks. Unlike static, principle-based frameworks, adaptive models integrate real-time monitoring of AI systems to assess fairness, transparency, and accountability throughout their lifecycle. For instance, dynamic bias-detection algorithms can continuously evaluate datasets and models for fairness, mitigating the risk of perpetuating inequities in patient care (Mehrabi et al., 2021). These adaptive frameworks should be co-designed with healthcare professionals to ensure they align with clinical workflows.

To overcome the trade-offs between transparency and model complexity, the adoption of hybrid models is essential. These models combine interpretable AI (e.g., decision trees) with high-performance "black-box" algorithms (e.g., neural networks). For instance, post-hoc explainability methods like SHAP (Shapley Additive Explanations) can provide localized insights into deep learning models, enabling clinicians to trust and understand AI predictions without sacrificing performance (Lundberg and Lee, 2017).

In addressing regulatory challenges like GDPR, a privacy-preserving AI approach such as federated learning offers a robust solution. Federated learning allows AI systems to train on decentralized datasets without transferring sensitive patient data, thereby maintaining privacy while enhancing model accuracy and fairness (Yang et al., 2019). This is particularly valuable for multi-institutional collaborations in healthcare.

Finally, implementing AI ethics boards within healthcare institutions can ensure accountability and oversight. These boards should include interdisciplinary stakeholders, such as clinicians, ethicists, and technologists, who evaluate the ethical implications of AI applications and provide actionable recommendations.

By integrating adaptive frameworks, hybrid models, privacy-preserving techniques, and ethics boards, healthcare systems can address current limitations and foster trust, transparency, and fairness in AI deployment.

Conclusion

Component one effectively demonstrated the use of machine learning techniques to predict global video game sales and identify hidden patterns in the data. Regression models and clustering algorithms, including K-Means and DBSCAN, provided valuable insights. Evaluation metrics validated the models, showcasing the importance of feature integration in sales forecasting and data segmentation.

This project demonstrated the application of Convolutional Neural Networks (CNNs) for image classification, emphasizing hyperparameter tuning and regularization techniques like dropout and batch normalization to enhance performance. The model effectively identified emergency vehicle images, achieving promising accuracy and validating the significance of optimization strategies in improving CNN efficacy for real-world applications.

Component Three highlights the critical ethical challenges in applying AI to healthcare, emphasizing the need for transparency, fairness, and trust. Addressing these challenges requires adaptive ethical frameworks, hybrid models, and privacy-preserving techniques. Interdisciplinary collaboration is essential to ensure AI systems are ethically deployed, fostering trust and improving healthcare outcomes globally.

Referencing

- Delua, J. (2021) Supervised vs. unsupervised learning: What's the difference? | IBM. www.ibm.com. Available online: https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning.
- Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R. and Chatila, R., 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities, and challenges toward responsible AI. Information Fusion, 58, pp.82-115.
- Holzinger, A., Biemann, C., Pattichis, C.S. and Kell, D.B., 2017. What do we need to build explainable AI systems for the medical domain?. arXiv preprint arXiv:1712.09923.
- Samek, W., Wiegand, T. and Müller, K.R., 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296.
- Tjoa, E. and Guan, C., 2020. A survey on explainable artificial intelligence (XAI): Towards medical XAI. IEEE Transactions on Neural Networks and Learning Systems, 32(11), pp.4793-4813.
- Topol, E., 2019. High-performance medicine: the convergence of human and artificial intelligence. Nature Medicine, 25(1), pp.44-56.
- European Commission, 2019. Ethics guidelines for trustworthy AI. European Commission.
- Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F. and Schafer, B., 2018. AI4People—An ethical framework for a good AI society: Opportunities, risks, principles, and recommendations. Minds and Machines, 28(4), pp.689-707.
- Lundberg, S.M. and Lee, S.I., 2017. A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.
- Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. and Galstyan, A., 2021. A survey on bias and fairness in machine learning. ACM Computing Surveys (CSUR), 54(6), pp.1-35.
- Yang, Q., Liu, Y., Chen, T. and Tong, Y., 2019. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2), pp.1-19.