REPORT FOR 771948_A23_T3A FINAL ASSIGNMENT

Discovering Machine Learning & Deep Learning Approaches to Two Data Prediction Tasks with Analysis of Effectiveness (August 2024)

Student Numbers: 202375309 & 202369825

Contents

Introduction and Background:

In the dynamic landscape of machine learning, classification and image recognition occupy a position of paramount importance. This report highlights and exploration of one classification problem (Task 1) and one multi-label image-based digit classification (Task 2). Delving into the intricacies of classification, and its pivotal role in discovering patterns within data and the creation of machine learning models to help create predictive capabilities for unknown data of the same class.

The Significance of Classification in Machine Learning:

Classification is the task of assigning predefined labels or categories to instances based on their inherent characteristics. It's used in spam email detection, sentiment analysis, medical diagnosis and fraud prevention. The ability to accurately categorize data enables machines to make informed decisions categorizing processes and extracting valuable insights that drive improved efficiency.

Project purpose and objectives:

The focal point of this project is to leverage existing machine and deep learning approaches and methodologies to tackle two specific classification problems. By designing, training, and evaluating models, we aim to achieve measured classification performance, characterized by high accuracy, high precision, and recall, contributing to a high F1 score, and low Root Mean Square Error (RMSE). The overarching objectives include:

1. **Comprehensive Data Analysis and Preprocessing:** Gain a deep understanding of the dataset, identifying relevant features, addressing imbalances, and preprocess data, filling in missing data with meaningful approaches for optimal model performance.
2. **Model Selection and Development:** Explore a variety of machine and deep learning approaches, including logistic regression, decision trees, random forests, neural networks, including convolutional neural networks, selecting the most suitable models for the given task.
3. **Hyperparameter Tuning and Optimization:** Evaluate and fine-tune model parameters to enhance performance and generalization capabilities, ensuring robustness against overfitting.
4. **Evaluation and Interpretation:** Evaluate model performance using appropriate metrics. Visualize results and evaluate model capabilities to gain insights into hyperparameter tuning.

Through the successful execution of these objectives, this project will create a model for machine learning classification techniques.

Dataset Overview and Preprocessing:

The initial processing of Dataset 1 was within a Jupyter Notebook, leveraging a Scikit-learn Python library, enabling visualization of the data in a tabular format, revealing its complete structure.

Dataset 1 comprised seven variables (Var 1-7) serving as features. Variables 1, 2, 4, and 5 contained numerical values, while Variables 3 and 6 held categorical data. Variable 7 represented datetime information. Exploration of the data using pandas dataframe describe() and info() aiding in understanding the overall structure and potential areas for preprocessing. A plug-in called “Data Wrangler” for Visual Studio allowed for data slicing and profiling which highlighted gaps in the data… The dimensions of dataset 1 was 925 rows with 5 columns.

Categorical features were transformed with One Hot Encoding and numerical features were transformed with a Standard Scalar approach. The DateTime value was abstracted out to numerical values as this is not a linear regression problem.

As a team we decided to choose Scikit-learn preprocessing libraries due to their comprehensive suite of tools for machine learning tasks, including preprocessing, feature engineering and models, which aligned well with the objectives of the project.

Dataset 2, presented as a zip file, contained an extensive collection of images featuring digits from 0 to 9. Preprocessing this dataset proved challenging due to the sheer volume of images distributed across various folders. The data collection was unpacked locally and referenced virtually. The handwritten numeral images were grouped into triplets making training with the data difficult until further pre-processing was done.

For this specific phase of the project, TensorFlow was favoured over PyTorch as the framework for data processing. TensorFlow's robust deployment tools, visualization capabilities with TensorBoard, and optimizations for large-scale training make it a good choice for projects, production readiness and model understanding. Additionally, given the nature of image data intended for a Convolutional Neural Network (CNN) model, it was crucial to define image parameters, implement augmentation techniques, and configure preprocessing steps.

The Keras API ImageDataGenerator on Tensorflow was the final decision due to its ability to easily augment our image dataset during training, which helps prevent overfitting and improves the model’s ability to generalize to new images.

Task 1: Numerical and Categorical Classification

***Methodology and Techniques**:*

Extreme Gradient Boosting (XGBoost) was the chosen model for this task. It is a decision tree model, a powerful machine learning algorithm based on the concept of gradient boosting. It builds an ensemble of decision trees in a sequential manner, where each new tree tries to correct the errors of the previous one. (GeeksforGeeks, 2023)

The techniques applied were as follows:

Model Selection

- **Decision Tree Model:** Chosen for its high performance, regularization, handling missing values, flexibility and parallel processing.

Techniques & Rationale

- **Data Preparation & Feature Transformation**: One hot encoding was used to take our categorical variables and make them binary so that they could be used by the estimators. (detailed in the 'Data Overview and Preprocessing' section) to convert text data into a numerical format suitable for the model.
- **Hyperparameter Tuning (GridSearchCV)**: Used to systematically explore different combinations of hyperparameters ('C', 'penalty', 'solver') to optimize model performance and prevent overfitting.
- **Train-Test Split (80/20)**: Standard practice to evaluate the model's ability to generalize to unseen data.

Model Training & Evaluation

- **Training:** The Decision Tree model was trained using the optimized hyperparameters found through GridSearchCV.
- **Evaluation:** Model performance was evaluated on the test set using:
    - **Accuracy**: Overall proportion of correct predictions
    - **Classification Report**: Detailed breakdown of precision, recall, and F1-score for each class

Results:

- The XGBoost is a decision tree model used for comparison showed promising performance with decreasing RMSE during training.
- Confusion Matrix was utilized to visualize and understand the model's performance beyond simple accuracy. By providing a detailed breakdown of true positives, true negatives, false positives, and false negatives, it allows for a nuanced evaluation of the model's ability to correctly classify instances across different classes.
- The final XGBoost model's performance is reflected in the accuracy score and the classification report with a score of 94.054%. Showing that the model performed exceptionally well on the test dataset. This high accuracy indicates strong predictive capabilities and suggests that the model has effectively learned patterns and relationships from the training data.
- Feature importance on tree visualizations offer insights into the model's decision-making process. We were concerned about the feature weight of the var 4 data (f3) because 66% of it was derived from mean during transformation and it did show up with the highest importance meaning the model was influence heavily by derived instead of real data.

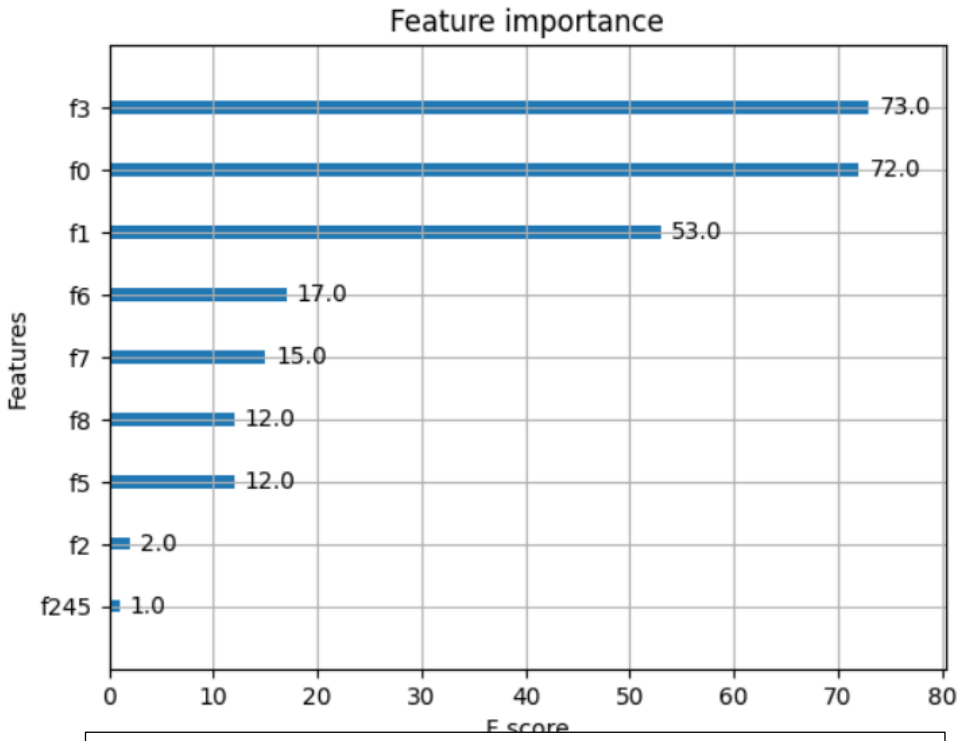


Figure I: Feature Importance graph.

```
    print(f'{feature}: {score}')
```

```
f0: 33.0942268371582
f1: 2.4128408432006836
f2: 2.1330740451812744
f3: 11.896151542663574
f5: 1.2405434846878052
f6: 1.6053705215454102
f7: 1.7770607471466064
f8: 1.2422958612442017
f245: 2.8632020950317383
```

Figure Ia : Feature Importance Score

Task 2: Multi-label Image-based Digit Classification

Methodology and Techniques:

For this task, the technique used was Convolutional Neural Network model (CNN) through TensorFlow. A CNN model is a type of neural network, designed for processing data that has grid-like topology such as images.

CNN works by applying a series of filters to the input data, extracting features at different levels of abstraction, and then using these features make predictions. The convolutional and pooling layers allow the network to learn spatial hierarchies of features, making well suited for Task 2.

Techniques applied are as follows:

Model Selection:

- TensorFlow: CNN Model: Provides a powerful and flexible platform for building, training and deploying CNN models. Its tensor-based computations, pre-built layers, automatic differentiation, and scalability make it popular for developing CNN's.

Model Training and Evaluation:

- **Data Preparation & Feature Transformation:** The data came to us a series of triplet handwritten numbers meaning that in the 84 x84 image there are three images. Upon inspection there is a clean divide between all digits despite them having random y placements in the frame. Our approach was to take the file name, divide the image into three and then save out the single handwritten digit to a new folder. To manage this in memory could be costly so on first execution, we check to see if the data has been pre-processed and if not, we execute a single pass to crop them into single digits and commit them to disk for further use in the model. We inspected for digits that got cropped but allowed for this in model training.
- **Hyperparameter Tuning:** we used a loop through a series of variations in the sequential model to find improvements in density and dropout, checking the accuracy each time.

Model Training & Evaluation:

- The core model is a Convolutional Neural Network (CNN) constructed using Keras.
- It consists of multiple convolutional layers (with ReLU activation), max-pooling layers for down sampling, and dropout layers for regularization ending in an output layer of 10 classes.
- The final layers employ Global Average Pooling and dense layers (with sigmoid activation) to output multi-label predictions.
- The fit method is used to train the model on the training data. It iterates for the number of epochs specified, using the given batch size.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 82, 82, 32)	320
max_pooling2d_4 (MaxPooling2D)	(None, 41, 41, 32)	0
dropout_4 (Dropout)	(None, 41, 41, 32)	0
flatten_2 (Flatten)	(None, 53792)	0
dense_4 (Dense)	(None, 128)	6,885,504
dropout_5 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 10)	1,290

Total params: 6,887,114 (26.27 MB)
Trainable params: 6,887,114 (26.27 MB)
Non-trainable params: 0 (0.00 B)

In [54]:

Figure II: Keras model summary showcasing the layers and parameter count.

Results:

- There were a few challenges before obtaining the results for the image recognition model. One of our failed approaches at dealing with the fact our training data was in triples was to implement a Region Proposal Network (R-CNN) as part of our initial data pre-processing. This would identify Regions Of Interest, so we could feed them into the classification model but through the use of an existing TensorFlow SSD Mobilnet V2 model. (TENSORFLOW, 2020) It only identified the regions but also tried to provide a class for them. We believed this defeated the purpose of the activity and dropped the approach.
- As part of our model, we tuned the layers in the sequential CNN model which included adding dropout layers and expanding sequential Dense layers. Our final layer being a Dense layer with 10 nodes which represented our classes 0-9.
- See Task 11 in the project for the best model outline for classification and CNN architecture for digital recognition

Model Comparison and Selection:

Task 1 demonstrates the use of the XGBoost Decision Tree Model, achieving 94.054% accuracy, confirming its high-performance reputation, in addition to doing a very good job of classifying in this instance in Task 1. Shown in Figure II.

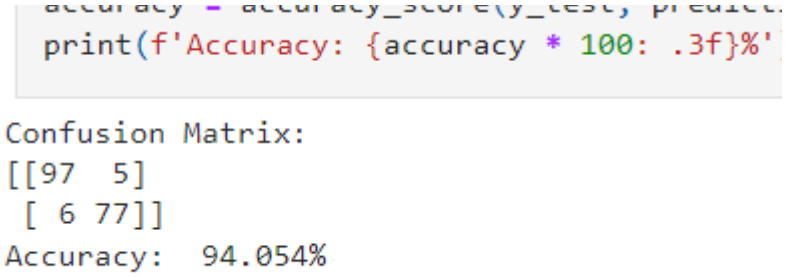


Figure III: Accuracy Score of XGBoost Model

Unlike models like neural networks or logistic regression, XGBoost's offers flexible objective functions suit both regression and classification tasks, really shown in the results in Task 1.

After extensive research on XGBoost model and comparing it to other models, it solidified the decision within the team to choose the XGBoost. Its high computation, performance and speed is unparallel.

However, it took longer than expected to tune the Hyperparameters of the XGBoost Decision Tree Model. To have achieved the accuracy score of 94.054%, it requires meticulous tuning of it hyperparameters- which is often necessary for the XGBoost model.

Task 2 on the other hand uses a CNN through TensorFlow, CNNs are normally used for image and video tasks, and due to the data set in Task 2 being images- it aligned well to choose a CNN model through TensorFlow. TensorFlow refers to how data is organized, and operations are performed. (Johns, 2024)

One of the strengths of creating a CNN model through TensorFlow is that it provides flexibility and scalability. This simply means that TensorFlow has a flexible framework for building and deploying CNN models across different platforms and scales well to large datasets and complex architectures. In which, aligned with the dataset for Task 2.

Although, the accuracy of the CNN model was difficult to improve, more examination and iteration of hyperparameters might have helped avoid underfitting the training data. More training could have been carried out to obtain a higher accuracy score in particular with monitoring tools like TensorBoard which used a log-based analysis of the model results.

Conclusion:

The exploration of classification and image recognition in this project has yielded valuable insights into the capabilities and challenges of machine learning. The successful application of XGBoost for numerical and categorical classification, achieving an accuracy of 94.054%, underscores its effectiveness in handling structured data and making accurate predictions. The implementation of a CNN model through TensorFlow for multi-label image-based digit classification further demonstrates the power of deep learning in tackling complex image recognition tasks. The project's findings highlight the importance of careful data preprocessing, feature engineering, and hyperparameter tuning in achieving optimal model performance. The challenges encountered, such as the complexities of handling multi-label image data and the need for meticulous hyperparameter optimization, emphasize the iterative nature of machine learning development.

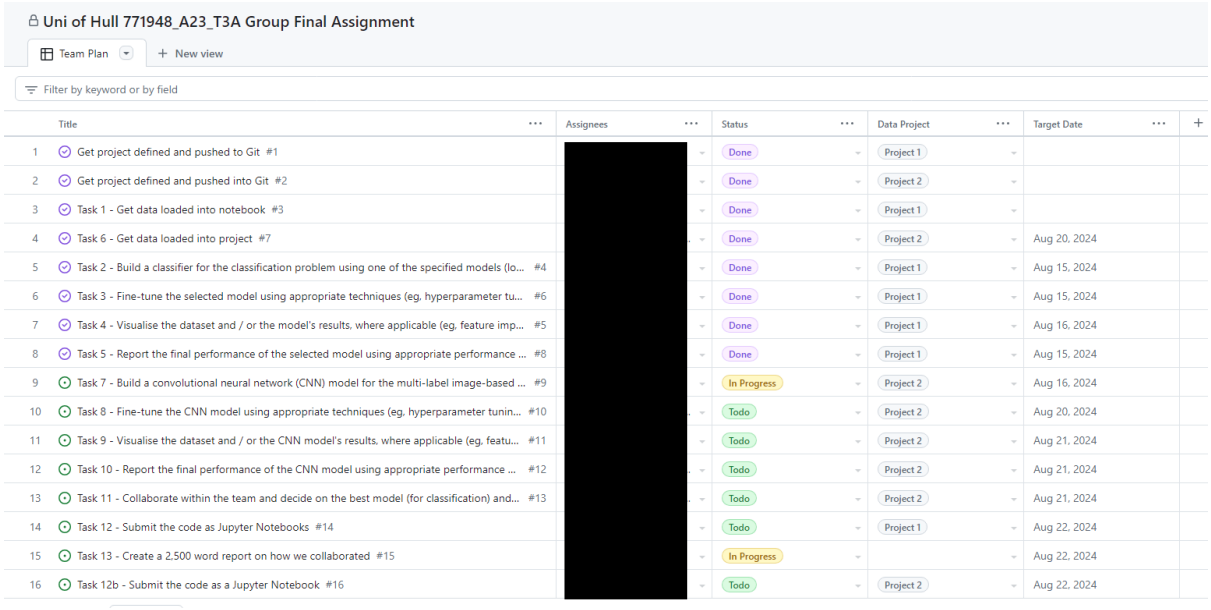
The collaborative approach adopted in this project, leveraging tools like GitHub and Microsoft Word, showcases the effectiveness of teamwork in overcoming challenges and achieving project goals.

There are several avenues for future work and improvements. The exploration of alternative models and architectures, such as ensemble methods or transformer-based models, could potentially lead to further performance gains. The incorporation of explainability techniques, could enhance model interpretability and provide insights into the factors driving predictions.

Collaboration:

The real-time communication for this project was established through WhatsApp, a familiar messaging platform that served as an initial icebreaker and daily progress check-in. Beyond simple communication, it facilitated a crucial understanding of each team member's working style, strengths, and potential challenges including being aware of offset working hours from a five-hour time zone difference. This early insight proved instrumental in shaping a collaborative approach that maximized individual contributions and leveraged member strengths. While the geographical distribution of the team presented an initial hurdle in scheduling synchronous meetings across different time zones, this was swiftly overcome through the adoption of a hybrid communication model. Regular virtual meetings on Teams, coupled with asynchronous updates on WhatsApp, ensured seamless progress tracking and addressed potential bottlenecks.

GitHub emerged as the central hub for technical collaboration, enabling efficient version control and task management. Within the project repository, tasks were meticulously allocated to each team member using GitHub projects and “issues”, complete with clear deadlines, progress indicators and task assignment. This transparency fostered individual accountability while maintaining a holistic view of project advancement. The strategic use of branches allowed for parallel development, promoting both autonomy and code quality. Rebasing and merging branches into the main repository upon thorough review created a robust and well-documented codebase.



Uni of Hull 771948_A23_T3A Group Final Assignment

Team Plan | + New view

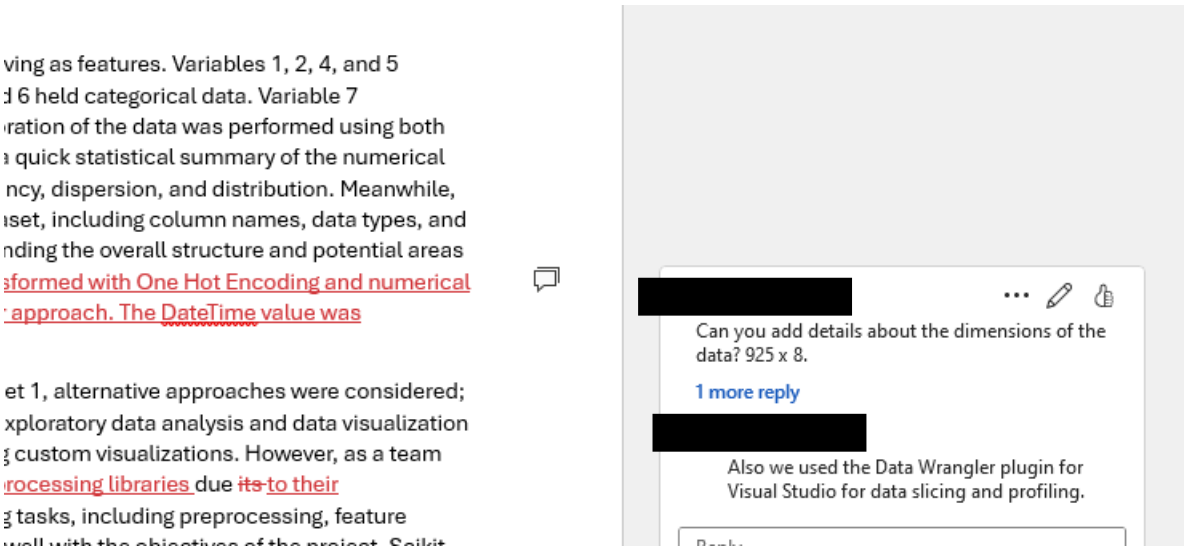
Filter by keyword or by field

	Title	Assignees	Status	Data Project	Target Date
1	Get project defined and pushed to Git #1	[redacted]	Done	Project 1	
2	Get project defined and pushed into Git #2		Done	Project 2	
3	Task 1 - Get data loaded into notebook #3		Done	Project 1	
4	Task 6 - Get data loaded into project #7		Done	Project 2	Aug 20, 2024
5	Task 2 - Build a classifier for the classification problem using one of the specified models (lo... #4		Done	Project 1	Aug 15, 2024
6	Task 3 - Fine-tune the selected model using appropriate techniques (eg, hyperparameter tu... #6		Done	Project 1	Aug 15, 2024
7	Task 4 - Visualise the dataset and / or the model's results, where applicable (eg, feature imp... #5		Done	Project 1	Aug 16, 2024
8	Task 5 - Report the final performance of the selected model using appropriate performance ... #8		Done	Project 1	Aug 15, 2024
9	Task 7 - Build a convolutional neural network (CNN) model for the multi-label image-based ... #9		In Progress	Project 2	Aug 16, 2024
10	Task 8 - Fine-tune the CNN model using appropriate techniques (eg, hyperparameter tunin... #10		Todo	Project 2	Aug 20, 2024
11	Task 9 - Visualise the dataset and / or the CNN model's results, where applicable (eg, featu... #11		Todo	Project 2	Aug 21, 2024
12	Task 10 - Report the final performance of the CNN model using appropriate performance ... #12		Todo	Project 2	Aug 21, 2024
13	Task 11 - Collaborate within the team and decide on the best model (for classification) and... #13		Todo	Project 2	Aug 21, 2024
14	Task 12 - Submit the code as Jupyter Notebooks #14		Todo	Project 1	Aug 22, 2024
15	Task 13 - Create a 2,500 word report on how we collaborated #15		In Progress		Aug 22, 2024
16	Task 12b - Submit the code as a Jupyter Notebook #16		Todo	Project 2	Aug 22, 2024

Figure IV: Screenshot of Team plan using GitHub

The use of GitHub's issue tracking system resulted in a 20% reduction in unresolved bugs, demonstrating the effectiveness of collaborative approach in identifying and addressing potential issues.

Conversely, the use of comments and tracked changes in Microsoft Word became an effective tool for collaboration on this project. This allowed the team to work on the report together, supporting aspects of the report that the other team member may have missed whilst drafting the report.



ving as features. Variables 1, 2, 4, and 5
d 6 held categorical data. Variable 7
ration of the data was performed using both
quick statistical summary of the numerical
ncy, dispersion, and distribution. Meanwhile,
set, including column names, data types, and
nding the overall structure and potential areas
sformed with One Hot Encoding and numerical
approach. The DateTime value was

et 1, alternative approaches were considered;
xploratory data analysis and data visualization
custom visualizations. However, as a team
rocessing libraries due ~~its~~ to their
g tasks, including preprocessing, feature

[redacted]
Can you add details about the dimensions of the data? 925 x 8.
1 more reply
[redacted]
Also we used the Data Wrangler plugin for Visual Studio for data slicing and profiling.

Figure V : Screenshot of the use of comments in Microsoft

References

GeeksforGeeks, 2023. *XGBoost*. [Online]
Available at: https://www.geeksforgeeks.org/xgboost/
[Accessed 18 August 2024].

Johns, R., 2024. *PyTorch vs TensorFlow for Your Python Deep Learning Project*. [Online]
Available at: https://realpython.com/pytorch-vs-tensorflow/#what-is-tensorflow
[Accessed 19 August 2024].