Assignment 3: Data modeling and Presentation report

Colonial Cancer Cell Spread Prediction

Practical Data Science (COSC2789)

Lecturer:

Dr. Thuy Nguyen

Team 2

Name	sID	Contribution %
Tran Phan Hoang Phuc	s3929597	33%
Bui Minh Nhat	s3878174	33%
Nguyen Xuan Thanh	s3915468	33%

Table of Contents

1. Abstract

Cancer is one of the most dangerous and life-threatening diseases in modern society, it is responsible for 1 out of every 6 deaths in 2021 worldwide, an alarming statistic about this disease[1]. As the mortality rate is quite significant and has an extremely high chance of growing over time, it's crucial to identify whether or not a patient has developed cancerous cells during the early stages of examination. This can create a major difference in the long-term survival rate for the patient as they can be accompanied for assistance and support. Considering this, the team has created and built a machine-learning model for classification to determine a patient's odds of developing or already having a cancer cell within their body. Inside the study the team has developed, we have created and tested three different classification models: traditional machine-learning models, XGBoost, LightGBM, and Voting Classifier. As well as a convolutional neural network model named RCC-Net.

2. Introduction

In 2022, it is estimated that more than 18.1 million cancer cases have been recorded worldwide[2]. Cancer prevention is one of the most crucial and foremost public health concerns of modern society due to its rapid growth rate and limitation in cure. Colorectal Cancer, or 'Colon' cancer for short, is a disease where cells within the rectum or colon of a human body multiply and grow at an uncontrollable rate[3]. It is one of the top three most popular forms of cancer worldwide[4]. The research question for this study is as follows:

"Which cell type might directly cause colon cancer? "

3. Methodology

3.1) Dataset

The dataset that was used for building and training the model is titled "Locality Sensitive Deep Learning for Detection and Classification of Nuclei in Routine Colon Cancer Histology Images" and was obtained from the United States National Library of Medicine website PubMed.gov[5]. The dataset consists of information for 60 different patients, with categorical features and details about the type of cell and whether or not it was cancerous. Divided into two sets, one main and one extra, it contains 6 data columns and over 20,000 data in total. More specifically, the data was comprised of: a unique patient's ID for each patient, the name of the image, the name of the cell type, encoded values for the name of the cell type, ID for each row, and binary values to determine if the cancer cell is cancerous or not (0 or 1).

3.1.1) Data Type

Dataset	Variable Name	Data Type
Categorical		
Main Dataset	isCancerous	Binary Nominal
	cellTypeName	Nominal
	cellType	Nominal (Encoded from cellTypeName)
Extra Dataset	isCancerous	Binary Nominal
Numerical		
Main Dataset	InstanceID	Discrete

Dataset	Variable Name	Data Type
	patientID	Discrete
Extra Dataset	InstanceID	Discrete
	patientID	Discrete

Table 1: Data Type Table

3.2) Data Analysis

Data cleaning is performed by using pathlib[6], Imagededup[7], and Python libraries such as pandas[8] and NumPy[9]. Initially, proofreading of the data included corrections for misspellings, typos, case inconsistency, outliers, and missing data points. The pathlib and Imagedeup packages were used to verify image data for duplication and corruption.

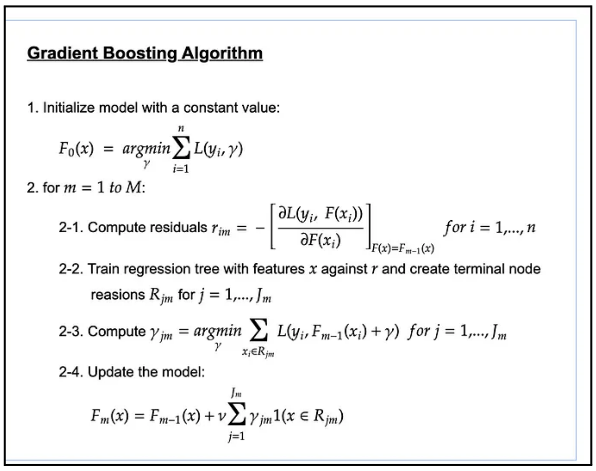
3.3) Data Visualization

The python plotting package Matplotlib[10], seaborn[11], plotly[12], sklearn[13] were used for data visualization.

3.4) Traditional Machine-Learning Models

3.4.1) XGBOOST

The first traditional machine-learning model the team decided to use was XGBoost. Boosting is a modeling approach combining several weak tree-based classifiers to create a high-performance classifier. In gradient boosting, each classifier corrects its predecessor error. Additionally, each classifier is trained using the residual errors of the predecessor as labels.



Gradient Boosting Algorithm

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma)$$

2. for $m = 1$ *to* M:

   2-1. Compute residuals $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ $for\ i = 1,\ldots,n$

   2-2. Train regression tree with features x against r and create terminal node reasions R_{jm} for $j = 1,\ldots,J_m$

   2-3. Compute $\gamma_{jm} = \underset{\gamma}{argmin} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$ $for\ j = 1,\ldots,J_m$

   2-4. Update the model:

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

Fig 1: Gradient Boosting Algorithm description

The XGBOOST Library implements the Gradient Boosting algorithm. It is a distributed, scalable gradient-boosted decision tree (GBDT) machine learning framework. It's considered one of the top machine learning libraries for regression, classification, and ranking issues with it offers of parallel tree boosting[14].

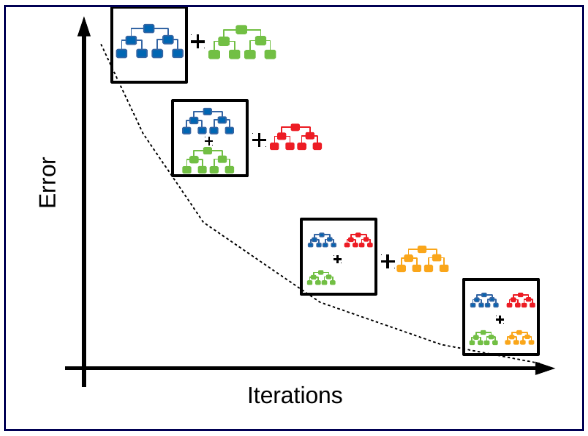


Fig 2: Gradient Boosting Illustrations

3.4.2) LightGBM

LightGBM (short for Light Gradient Boosting Machine) is a framework that uses tree-based learning algorithms [15]. It grows the decision tree leaf-wise (vertically) which differs from other boosting algorithms where it grows depth-wise (horizontally) [16].

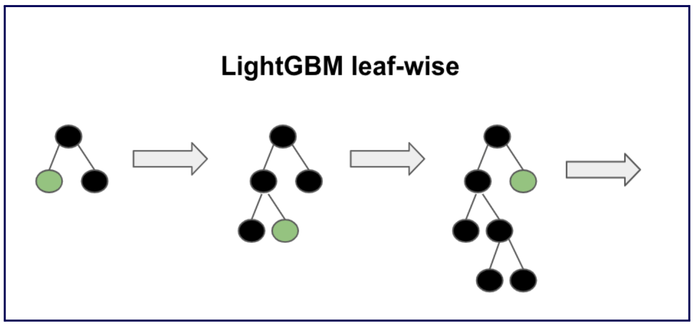


Fig 3: LightGBM leaf-wise grow

It can handle large datasets and is faster than other popular libraries (ex: XGBoost).

LightGBM includes unique features [17]:

+ Support for categorical features
+ Efficient handling of missing values
+ Support for parallel processing.

3.5) Convolutional Neural-Network Machine-Learning Model

A convolutional neural network (CNN) is an artificial neural network for numerous applications and data types. CNN is a type of deep learning model utilized for image recognition and pixel data processing[18].

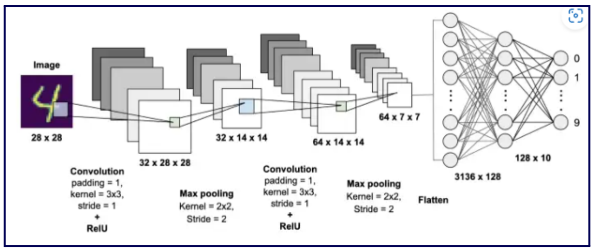


Image
28 x 28
32 x 28 x 28
32 x 14 x 14
64 x 14 x 14
64 x 7 x 7
3136 x 128
128 x 10
0
1
9
Convolution padding = 1, kernel = 3x3, stride = 1 + RelU
Max pooling Kernel = 2x2, Stride = 2
Convolution padding = 1, kernel = 3x3, stride = 1 + RelU
Max pooling Kernel = 2x2, Stride = 2
Flatten

Fig 4: CNN architecture

3.5.1) RCC_Net

RCC_Net is a deep learning model (CNN-based architecture). RCC_Net reduces dimensions to compress and extract key features[19]. The fundamental objective of this network is to make the CNN model as basic as possible. In terms of training time, it is likewise significantly more efficient than deeper and more complicated networks[20].

3.6) Voting Classifiers

A voting classifier is a machine learning model that gains experience by training on a collection of several models and forecasts an output (class) based on the class with the highest likelihood of becoming the output. To forecast or classify the output class based on the largest majority of votes, it averages the results of each classifier provided into the voting classifier.

The concept is to build a single model that learns from various models and predicts output based on their aggregate majority of votes for each output class rather than building separate specialized models and determining their accuracy. In this project, scikit- learn's Voting Classifiers are used due to their compatibility with XGBoost and LightGBM[21].

3.7) Hyperparameter Optimization

3.7.1) Approaches

To tune the hyperparameter of the Machine Learning Algorithms, three different approaches were used:

a) **Optuna**: Use for automatic extraction of parameters when tuning of XGBoost and LightGBM.
b) **GridSearchCV**: Use for extracting parameters when tuning the Ensemble model.
c) **Early Stopping**: Use for tuning CNN model

Optuna is a machine learning-specific automated hyperparameter tuning software framework that may be used with other frameworks such as PyTorch, TensorFlow, Keras, SKlearn, etc. Optuna offers a feature known as define-by-run API to assist users in creating highly modular code and search areas for hyperparameters on the fly[22]. Under the hood, Optuna implements the Bayesian optimization algorithm (TPE).

Bayesian optimization iteratively creates a probabilistic model of the function mapping from hyperparameter values to the goal function. The Bayesian optimization techniques look for global optimization to create a posterior distribution over the objective function. The probabilistic model beliefs are collected based on the behavior of the function [23].

A TPE Bayesian optimization algorithm randomly selects a subset of hyperparameters and sorts them based on their scores initially. After selecting and sorting, The hyperparameters are further divided into two groups based on some predefined quantile. The two groups are then modeled into estimated densities using Parzen kernel density estimators. After the location of the hyperparameters with the highest expected improvement, the hyperparameter with the highest improvement expectation is verified, sorted, and split iteratively. The whole process will be executed until the Budget is finished and returns the most optimal hyperparameters [24].

3.7.2) Hyperparameters framework

The rationale behind the parameters tuning functions/frameworks is as follow:

Optuna*:* Optuna is widely used in the data science community due to its performance and intuitive dashboard.

GridSearchCV*:* the ensemble has only two tunable parameters (voting and weights) and is a reasonable length.

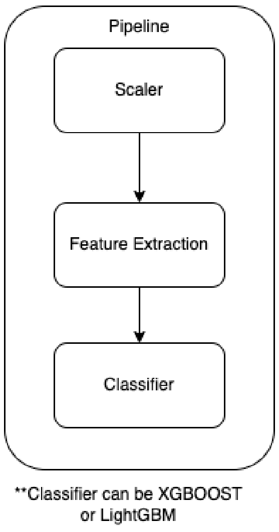


Fig 5: Optuna hyperparameter tuning setup

For the hyperparameter tuning, the following setup is tuned using GridSearchCV:

Since in the training phase of LightGBM Classifier and XGBoost Classifier, MinMax scaler yields the highest result. Therefore it was selected for building the training pipeline of the Ensemble models. The data extraction step was not included in the pipeline since the model performed better without any feature extraction, according to the result of Optuna.

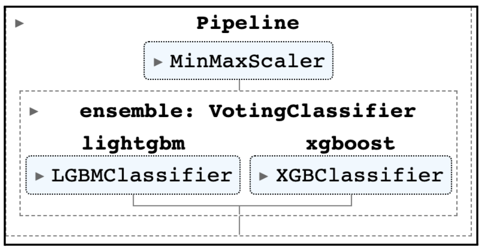


Fig 6: Ensemble Pipeline setup

To view the Optuna recommended extracted parameters, please refer to table 9 in the appendix

By using `trial.suggest` function, Optuna will automatically suggest a parameter within the specified range. GridSearchCV is used to tune the following hyperparameter following [] recommendations:

Process Name	***Parameters***
Voting Classifier	*'ensemble__voting': ['soft'], 'ensemble__weights' : [[1,2],[2,1],[1,1]]*

Table 2: Voting Classifier hyperparameters tuning.

3.8) Model Evaluation

The parameters to evaluate the performances of the machine-learning model can be determined through the following formula(s).

Parameters	***Formula***
$Classification\ error\ rate$	$\frac{(FP+FN)}{(TP+TN+FP+FN)}$

Classification accuracy	$\frac{(TP+TN)}{(TP+TN+FP+FN)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F1 measure	$2\times\frac{(precision\times recall)}{(precision+recall)}$

Table 3: Model Evaluation Formula Table

To view the definition for the variables, please refer to table 10 in the appendix

4) Data Cleaning & Exploration

4.1) Preprocessing EDA

Hypotheses to be examined (apply for before and processing the data):

- **Hypothesis 1:** Cell type 1 have the highest frequency
- **Hypothesis 2:** Cancerous cells account for over 50%

Hypothesis 1: Cell type 2 have the highest frequency

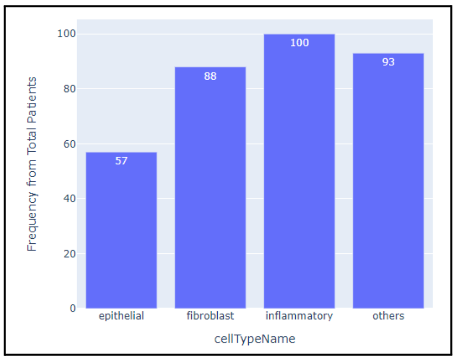


Fig 7: Cell type frequency from total patients

As seen from the histogram, cell type 1 does have the highest frequencies out of the data. With the least being cell type 3. This supports the initial hypothesis of 'Cell type 1 has the highest frequency'.

Hypothesis 2: Cancerous cells account for over 50%

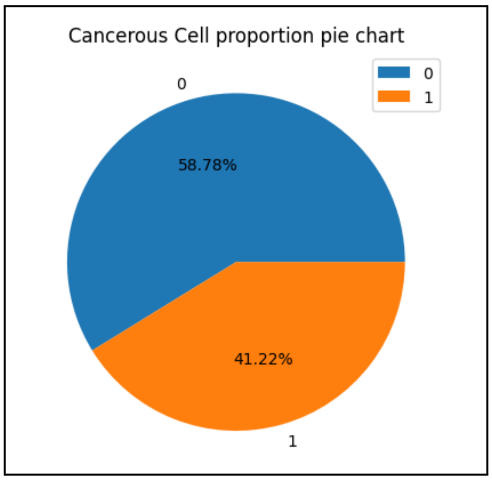


Fig 8: Cancerous cell proportion

The bar plot provides us with a clearer view of which patient (identified with patient ID) has cancerous cells. And as observed, a lot of the patients don't have any cancerous cells (0).

As observed from the pie chart, the percentage of those that do not have cancerous cells is higher than those without. The rate of those in the data that have cancerous cells is only around 41%. This disproves the initial hypothesis that "Cancerous cells account for over 50%".

4.2) Data Cleaning

The practice of correcting or deleting inaccurate, damaged, improperly formatted, duplicated, or missing data from a dataset is known as data cleaning. There are several ways for data to be duplicated or incorrectly categorized when merging different data sources. Even though results and algorithms appear correct, they are unreliable if the data is inaccurate [25].

To ensure the performance of model the following step was perform for data cleaning:

Steps	Task
1	Check and adjusting the trailing white space
2	Checking and adjusting case inconsistencies
3	Checking and correcting the NaN value
4	Checking and adjusting corrupted image
5	Checking and removing duplicate image content by verifying the hash of the image

Table 4: Data Cleaning steps table

4.3) Postprocessing EDA

Hypotheses to be examined (apply for after processing the data):

- **Hypothesis 1:** Cell type 2 is the only cancerous cell
- **Hypothesis 2:** Cell type 1 have the highest frequency
- **Hypothesis 3:** Cancerous cells account for over 50%
- **Hypothesis 4:** Image being identified as cancerous for each patientID will be over 50%

Hypothesis 1: Cell type 2 is the only cancerous cell

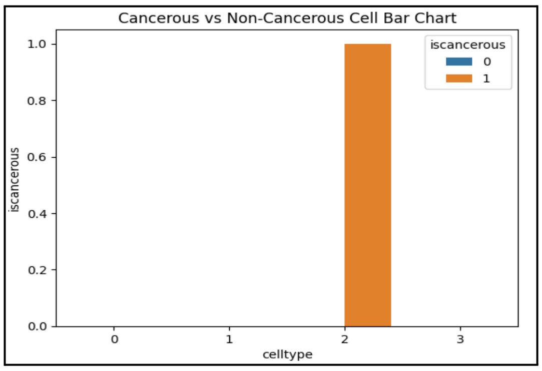


Fig 9: Cancerous cell vs non cancerous cell bar chart

As we observed from the dataset, cell type 2 is the only cell that gets cancerous since cell type 2 got the sum from all values "1" (which means "True" in "iscancerous" column). Other cells get 0 since all their values got "0" (which means "False" in "iscancerous" column).

Hypothesis 2: Cell type 2 has the highest frequency

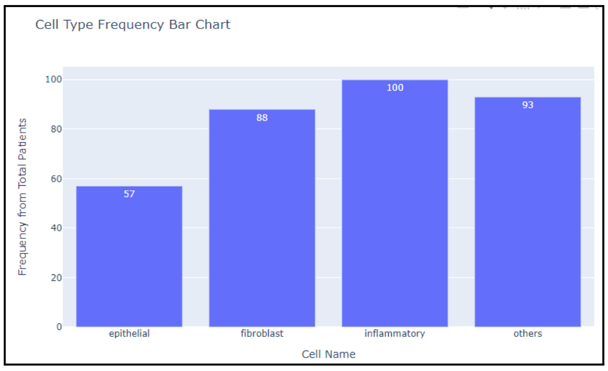


Fig 10: Cell Type Frequency Barchart

As seen from the histogram, cell type 2 does have the highest frequencies out of the data. With the least being cell type 3. This supports the initial hypothesis of ‘Cell type 2 has the highest frequency’.

Hypothesis 3: Cancerous cells account for over 50%

As stated in the justifications for the previous two hypotheses, cell type 2 is the sole malignant cell (hypothesis 2.1) and covers 57% of patients (hypothesis 2.2). As a result, 57% of patients with cell 2 type have cancer.

Hypothesis 4: The image identified as cancerous for each patientID will be over 50%

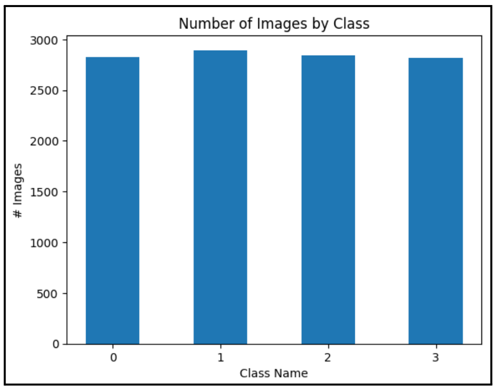


Fig 11: Number of images by class

As indicated previously, the images were already labeled as cancerous or not. However, classifying images by classes also involves exploring image data and its structure. The bar chart revealed that there were around 2500 photos for each cell type.

Since the dataset already contains cancerous image labels and is based on the three preceding hypotheses:

+ Cell 2 is the cancerous cell (hypothesis 2.1)

+ Cell 2 accounts for 57% of total patients (hypothesis 2.2)

+ 57% of total patients have cancer (hypothesis 2.3)

Consequently, images of cell 2 are cancerous images which mean 57% of the total images are identified as cancerous

5) Feature Engineering

5.1) Feature engineering procedures

Feature engineering is the process of developing, transforming, extracting, and choosing features—also referred to as variables—that are most suited for a well performed ML algorithm.

These procedures include

- **Feature creation**: Determining the factors that will be most helpful in the prediction model is the first step in developing features. Existing features are combined through addition, subtraction, multiplication, and ratio to develop new, more powerful derived features.
- **Transformation:** To improve model performance, transformation entails changing the predictor variables. This is done to ensure all variables are on the same scale to make the model easier to understand, that accuracy is improved, and that all features are within an acceptable range for the model to avoid computational errors.
- **Feature extraction:** Feature extraction involves creating new variables by feature extraction from unprocessed data. Automatically condensing the amount of data into a more manageable modeling set is this stage's goal.
- **Feature Selection:** The process of feature selection is to evaluate, appraise, and rank different characteristics to determine which aspects are essential for the model and should be prioritized, which features are redundant and should be deleted, and which features should be irrelevant and discarded.

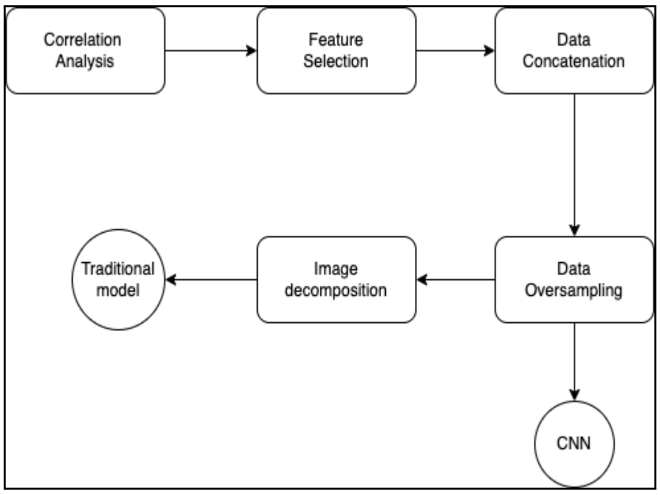
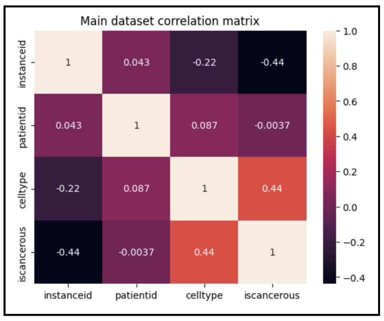
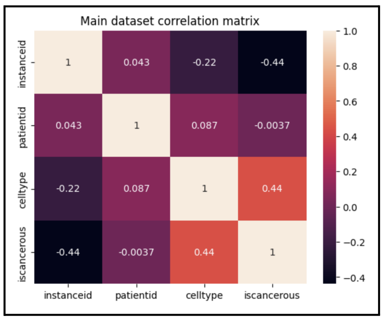
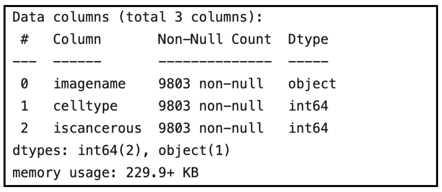
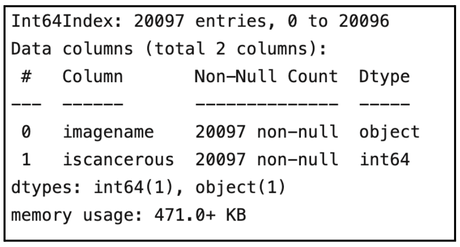


Fig 12: Feature engineering procedure

Steps	Figure	Description
Correlation Analysis	 **Fig 13: Main dataset correlation analysis**	In the main dataset, we discover that 'celltype' and 'patientid' are correlated with the 'iscancerous' column with the Pearson correlation coefficient value of 0.44 and - 0.44 respectively. However the 'patientid' and 'instanceid' are used for identification purposes only. For the extra dataset, both the 'patientid' and 'instance id' are correlated with 'iscancerous' columns with the Pearson correlation coefficient of -0.37 and -0.35 respectively. However, the 'patientid' and 'instance id' only serve identification purposes.

	instanceid	patientid	celltype	iscancerous
instanceid	1	0.043	-0.22	-0.44
patientid	0.043	1	0.087	-0.0037
celltype	-0.22	0.087	1	0.44
iscancerous	-0.44	-0.0037	0.44	1

	 **Fig 14: Extra dataset correlation matrix**	

Feature Selection	
	`Data columns (total 3 columns):`
	` #   Column       Non-Null Count  Dtype`
	`---  ------       --------------  -----`
	` 0   imagename    9803 non-null   object`
	` 1   celltype     9803 non-null   int64`
	` 2   iscancerous  9803 non-null   int64`
	`dtypes: int64(2), object(1)`
	`memory usage: 229.9+ KB`
	Fig 15: Feature selection for the main dataset.
	
	`Int64Index: 20097 entries, 0 to 20096`
	`Data columns (total 2 columns):`
	` #   Column       Non-Null Count  Dtype`
	`---  ------       --------------  -----`
	` 0   imagename    20097 non-null  object`
	` 1   iscancerous  20097 non-null  int64`
	`dtypes: int64(1), object(1)`
	`memory usage: 471.0+ KB`
	Fig 16: Feature selection for the extra dataset.
		In the main dataset, the 'imagename', 'iscancerous' and 'celltype' are selected. The remaining columns are dropped to prevent data leakage.
		For the extra dataset, the 'iscancerous' and 'imagename' are selected from the extra dataset.

Data concatenation	**N/A**	Both extra and main datasets have 'iscancerous', and 'imagename'. A new dataset is created by concatenating two features of the two datasets. Thus increasing the number of data samples for the classification of 'iscancerous' variables improves the classifier's performance.
Oversampling	**To view oversampling pie charts, please refer to table 11 in the appendix**	After finishing the data concatenation procedure, the class distribution of the class 'iscancerous' for task 1 and 'celltype' for task 2 is examined. According to … if the dataset is not correctly balanced, the classifiers might develop a bias for one class. This could cause a reduction in the model performance. To resolve this issue, Synthetic Minority Oversampling Technique, or SMOTE, is used for balancing the class in both datasets as it is a widely used technique.
Image decomposition		To make our image dataset compatible with the XGBoost, LightGBM, and Voting Classifier, we decompose images using np.ravel method into features vectors.

Table 5: Feature Engineering procedures table

6) Prediction Modelling

The model was trained in the following configuration**:**

+ **Python:** Python version 3.9
+ **Operation System:** Windows 11/ macOS
+ **Anaconda Navigator version:** 2.3.2
+ **Training epochs (if any):** 100

Task 1	
Model Name	**Accuracy Score**
XGBoost Standard	0.915
XGBoost Tuned	0.926
LightGBM Standard	0.912
LightGBM Tuned	0.924
Voting Classifier (LightGBM + XGBoost)	0.927
RCC_Net	0.927

Table 6: Task 1 accuracy score table

Task 2	
Model Name	**Accuracy Score**
XGBoost Standard	0.908
XGBoost Tuned	0.911
LightGBM Standard	0.904
LightGBM Tuned	0.912
Voting Classifier (LightGBM + XGBoost)	0.911
RCC_Net	0.876

Table 7: Task 1 accuracy score table

The details result and parameters could be found under appendix section in table 12 and table 13

7) Discussion

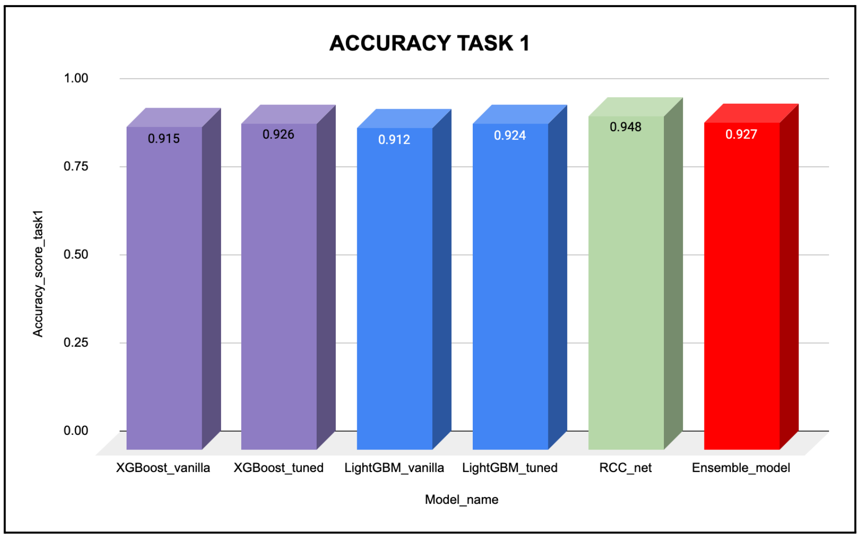


Fig 17: Task 1 model accuracy graph

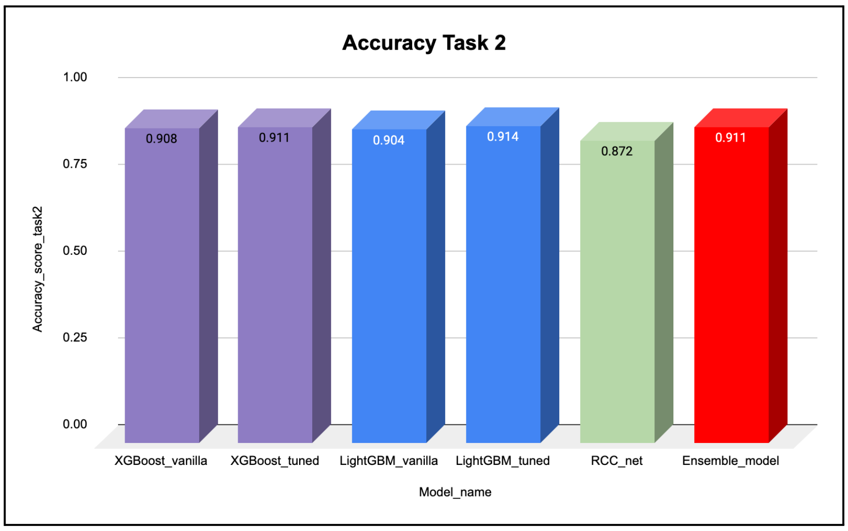


Fig 18: Task 2 model accuracy graph

As observed for both task 1 and tak 2 accuracy scores for every model the team has built yields the following results:

+ The machine-learning model with the highest accuracy score for Task 1 is: RCC_Net
+ The machine-learning model with the highest accuracy score for Task 2 is: LightGBM_tuned

8) Conclusions

Since the research question seeks to identify the primary cell type that causes colon cancer, the responses were demonstrated during the project. Initially, in the EDA post-processing first hypothesis, cell type 2 was identified as the sole cancerous cell type. RCC Net (task 1), with a 94.1% accuracy score, and LightGBM tuned (task 2), with a 91.4% accuracy score, are the two models with the highest accuracy score following the model-building process. Overall, this evidence was sufficient to establish that cell type 2 is the primary colon cancer-causing cell.

References

[1] "How many people die of cancer a year?," WebMD. [Online]. Available: https://www.webmd.com/cancer/how-many-cancer-deaths-per-year. [Accessed: 19-Jan-2023].

[2] World Cancer Research Fund International, "Worldwide cancer data: World cancer research fund international," WCRF International, 14-Apr-2022. [Online]. Available: https://www.wcrf.org/cancer-trends/worldwide-cancer-data/. [Accessed: 17-Jan-2023].

[3] "What is colorectal cancer?," Centers for Disease Control and Prevention, 17-Feb-2022. [Online]. Available: https://www.cdc.gov/cancer/colorectal/basic_info/what-is-colorectal-cancer.htm#:~:text=Colorectal%20cancer%20is%20a%20disease,the%20colon%20to%20the%20anus. [Accessed: 17-Jan-2023].

[4] Worldwide cancer data: World cancer research fund international. WCRF International. (2022, April 14). Retrieved January 17, 2023, from https://www.wcrf.org/cancer-trends/worldwide-cancer-data/#:~:text=Breast%20and%20lung%20cancers%20were,contributing%2010.7%25%20of%20new%20cases

[5] K. Sirinukunwattana, S. E. A. Raza, Y. Tsang, D. R. J. Snead, I. A. Cree and N. M. Rajpoot, "Locality Sensitive Deep Learning for Detection and Classification of Nuclei in Routine Colon Cancer Histology Images," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1196

[6] "Pathlib - object-oriented filesystem paths," Python documentation. [Online]. Available: https://docs.python.org/3/library/pathlib.html. [Accessed: 19-Jan-2023].

[7] idealo D. S. Team, "Image deduplicator (imagededup)," Imagededup. [Online]. Available: https://idealo.github.io/imagededup/. [Accessed: 19-Jan-2023].

[8] "Pandas," pandas. [Online]. Available: https://pandas.pydata.org/. [Accessed: 19-Jan-2023].

[9] NumPy. [Online]. Available: https://numpy.org/. [Accessed: 19-Jan-2023].

[10] "Visualization with python," Matplotlib. [Online]. Available: https://matplotlib.org/. [Accessed: 19-Jan-2023].

[11] "Statistical Data Visualization#," seaborn. [Online]. Available: https://seaborn.pydata.org/. [Accessed: 19-Jan-2023].

[12] "Plotly," Plotly Python Graphing Library. [Online]. Available: https://plotly.com/python/. [Accessed: 19-Jan-2023].

[13] "Learn," scikit. [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 19-Jan-2023].

[14] "What is XGBoost?," NVIDIA Data Science Glossary. [Online]. Available: https://www.nvidia.com/en-us/glossary/data-science/xgboost/. [Accessed: 19-Jan-2023].

[15] S. Shisingh, "Lightgbm (Light Gradient Boosting Machine)," GeeksforGeeks, 22-Dec-2021. [Online]. Available: https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/. [Accessed: 17-Jan-2023].

[16] S. Surana, “What is light GBM? advantages &amp; disadvantages? LightGBM vs XGBoost?: Data Science and Machine Learning,” Kaggle, 2021. [Online]. Available: https://www.kaggle.com/general/264327. [Accessed: 17-Jan-2023].

[17] G. Ke, “Features,” Features - LightGBM 3.3.3.99 documentation. [Online]. Available: https://lightgbm.readthedocs.io/en/latest/Features.html. [Accessed: 17-Jan-2023].

[18] R. Awati, “What are convolutional neural networks?,” Enterprise AI, 29-Sep-2022. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network. [Accessed: 19-Jan-2023].

[19] “Deep residual coalesced convolutional network for efficient semantic ...” [Online]. Available: https://www.researchgate.net/publication/315136454_Deep_residual_coalesced_convolutional_network_for_efficient_semantic_road_segmentation. [Accessed: 19-Jan-2023].

[20] “Global survey,” arXiv.org e-Print archive. [Online]. Available: https://arxiv.org/. [Accessed: 19-Jan-2023].

[21] “ML: Voting classifier using Sklearn,” GeeksforGeeks, 25-Nov-2019. [Online]. Available: https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/. [Accessed: 19-Jan-2023].

[22] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” arXiv.org, 25-Jul-2019. [Online]. Available: https://arxiv.org/abs/1907.10902. [Accessed: 19-Jan-2023].

[23] Y. Lim, “State-of-the-art machine learning hyperparameter optimization with optuna,” Medium, 01-Apr-2022. [Online]. Available: https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1. [Accessed: 19-Jan-2023].

[24] Y. Lim, “State-of-the-art machine learning hyperparameter optimization with optuna,” Medium, 01-Apr-2022. [Online]. Available: https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1. [Accessed: 19-Jan-2023].

[25] “Guide to data cleaning: Definition, benefits, components, and how to clean your data,” Tableau. [Online]. Available: https://www.tableau.com/learn/articles/what-is-data-cleaning#:~:text=Data%20cleaning%20is%20the%20process,to%20be%20duplicated%20or%20mislabeled. [Accessed: 19-Jan-2023].

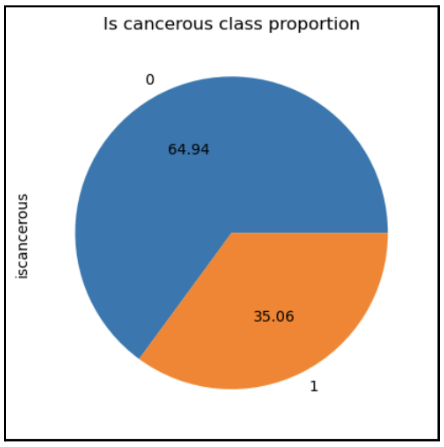
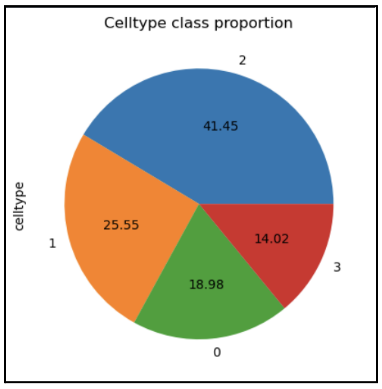
Appendix

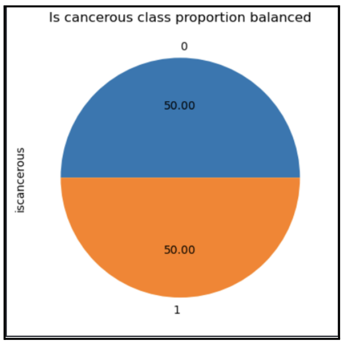
Recommended Parameters Table	
Process Name	**Parameters**
Scaler	"scalers", ['minmax', 'standard', 'robust']
Feature Extraction	"dim_red", ["PCA", None]
Classifier (XGBoost)	'xgboost__max_depth': trial.suggest_int('max_depth', 1, 9), 'xgboost__learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 1.0), 'xgboost__n_estimators': trial.suggest_int('n_estimators', 50, 500), 'xgboost__min_child_weight': trial.suggest_int('min_child_weight', 1, 10), 'xgboost__gamma': trial.suggest_loguniform('gamma', 1e-8, 1.0), 'xgboost__subsample': trial.suggest_loguniform('subsample', 0.01, 1.0), 'xgboost__colsample_bytree': trial.suggest_loguniform('colsample_bytree', 0.01, 1.0), 'xgboost__reg_alpha': trial.suggest_loguniform('reg_alpha', 1e-8, 1.0), 'xgboost__reg_lambda': trial.suggest_loguniform('reg_lambda', 1e-8, 1.0), 'xgboost__eval_metric': 'mlogloss', 'xgboost__use_label_encoder': False
Classifier (LightGBM)	"lgbm__metric": "mlogloss", "lgbm__verbosity": -1, "lgbm__boosting_type": "gbdt", "lgbm__lambda_l1": trial.suggest_float("lambda_l1", 1e-8, 10.0, log=True), "lgbm__lambda_l2": trial.suggest_float("lambda_l2", 1e-8, 10.0, log=True), "lgbm__num_leaves": trial.suggest_int("num_leaves", 2, 256), "lgbm__feature_fraction": trial.suggest_float("feature_fraction", 0.4, 1.0), "lgbm__bagging_fraction": trial.suggest_float("bagging_fraction", 0.4, 1.0), "lgbm__bagging_freq": trial.suggest_int("bagging_freq", 1, 7), "lgbm__min_child_samples": trial.suggest_int("min_child_samples", 5, 100),

Table 9: Recommended Parameters Table

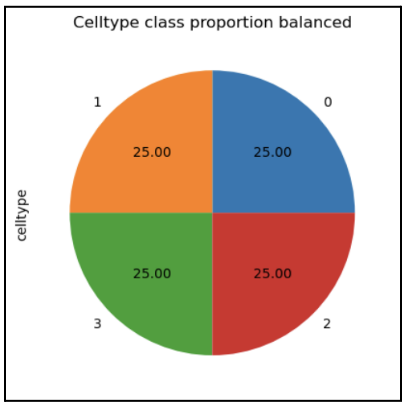
Variables Definition table	
Variable	**Definition**
FP	**The number of false positives (predicted positives that were actually negative instances)**
FN	**The number of false negatives (predicted negatives that were actually positive instances)**
TP	**The number of true positives**
TN	**The number of true negatives**

Table 10: Variables Definition table

Oversampling charts Figs
 **Class distribution of iscancerous variable**  **Class distribution for cell type variable**



iscancerous class after SMOTE

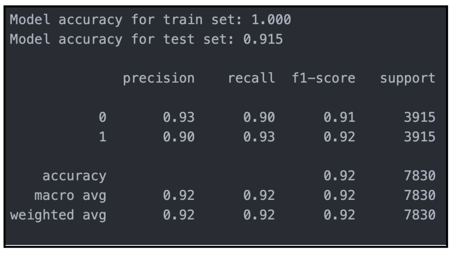


celltype after SMOTE

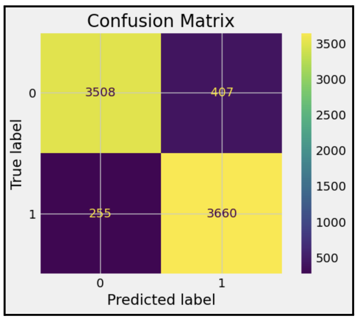
Table 11: Oversampling figures tables

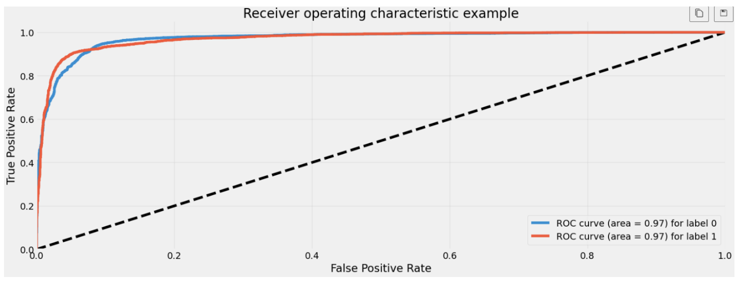
Task 1 Model Result		
Model Name	Score	Matrix & Curve

XGBOOST Standard

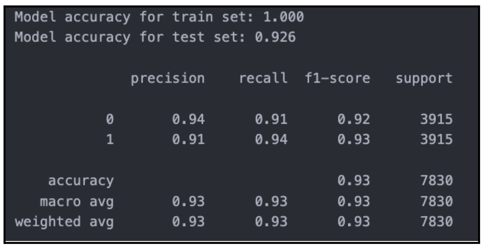


```
Model accuracy for train set: 1.000
Model accuracy for test set: 0.915

              precision    recall  f1-score   support

           0       0.93      0.90      0.91      3915
           1       0.90      0.93      0.92      3915

    accuracy                           0.92      7830
   macro avg       0.92      0.92      0.92      7830
weighted avg       0.92      0.92      0.92      7830
```

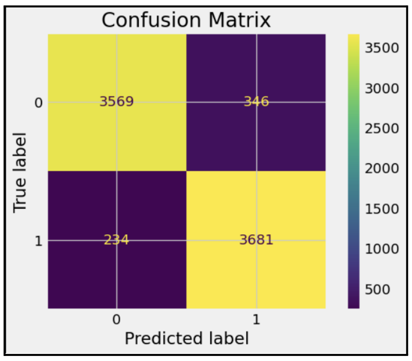


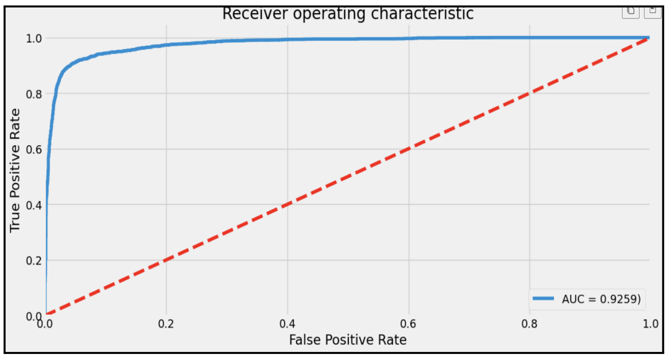


XGBoost Tuned

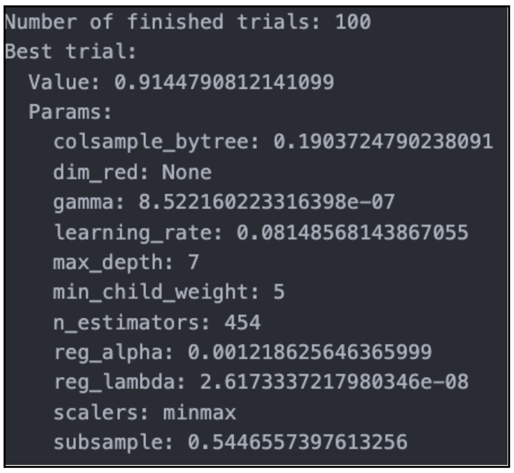


```
Model accuracy for train set: 1.000
Model accuracy for test set: 0.926

              precision    recall  f1-score   support

           0       0.94      0.91      0.92      3915
           1       0.91      0.94      0.93      3915

    accuracy                           0.93      7830
   macro avg       0.93      0.93      0.93      7830
weighted avg       0.93      0.93      0.93      7830
```



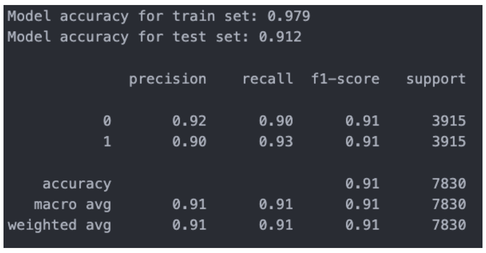


BEST PARAMETERS FOR XGBoost

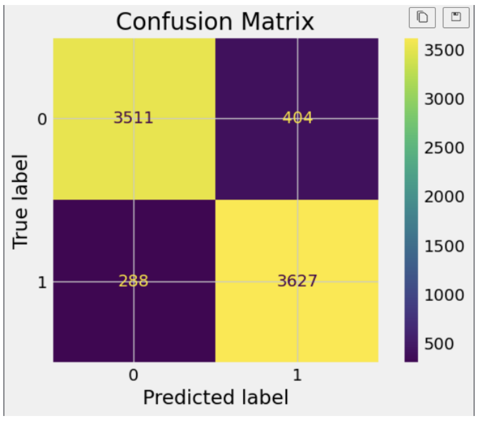


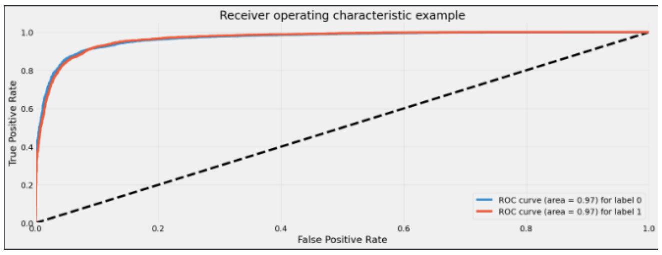
```
Number of finished trials: 100
Best trial:
  Value: 0.9144790812141099
  Params:
    colsample_bytree: 0.1903724790238091
    dim_red: None
    gamma: 8.522160223316398e-07
    learning_rate: 0.08148568143867055
    max_depth: 7
    min_child_weight: 5
    n_estimators: 454
    reg_alpha: 0.001218625646365999
    reg_lambda: 2.6173337217980346e-08
    scalers: minmax
    subsample: 0.5446557397613256
```

LightGBM

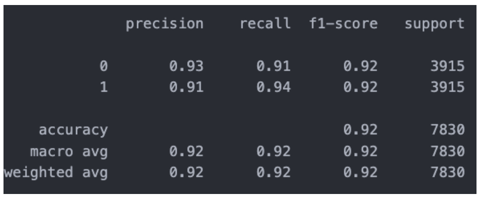


```
Model accuracy for train set: 0.979
Model accuracy for test set: 0.912

              precision    recall  f1-score   support

           0       0.92      0.90      0.91      3915
           1       0.90      0.93      0.91      3915

    accuracy                           0.91      7830
   macro avg       0.91      0.91      0.91      7830
weighted avg       0.91      0.91      0.91      7830
```

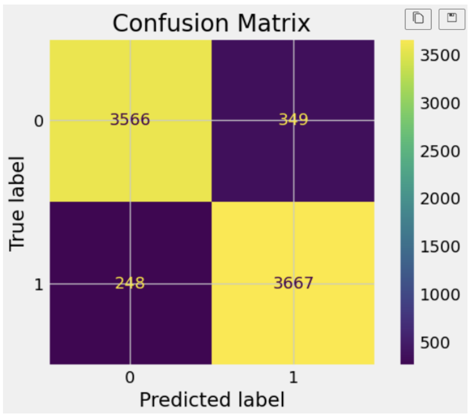


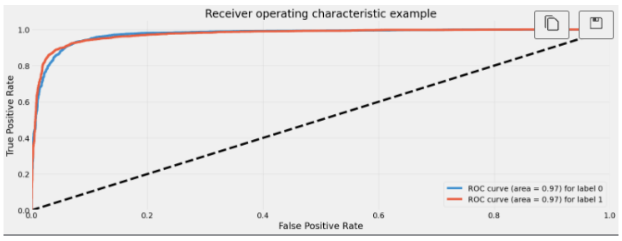


LightGBM Tuned

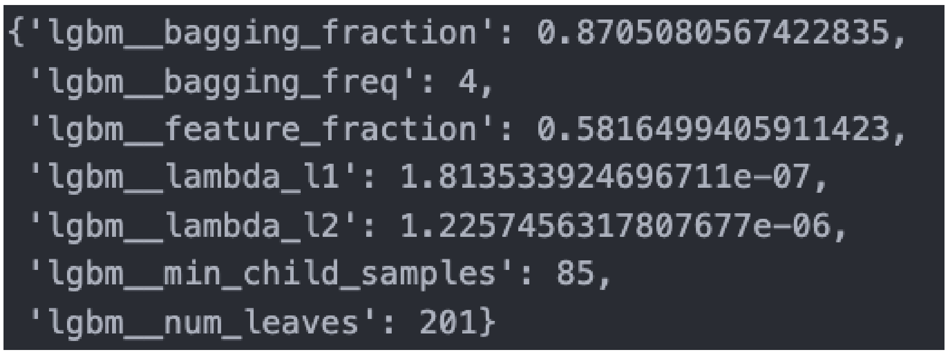


```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      3915
           1       0.91      0.94      0.92      3915

    accuracy                           0.92      7830
   macro avg       0.92      0.92      0.92      7830
weighted avg       0.92      0.92      0.92      7830
```



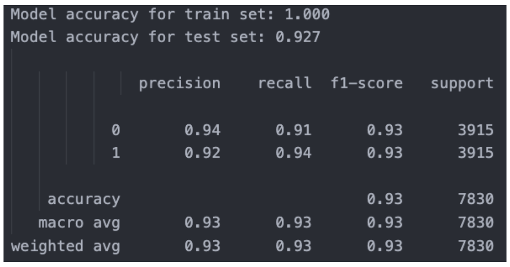


BEST PARAMETERS FOR LightGBM

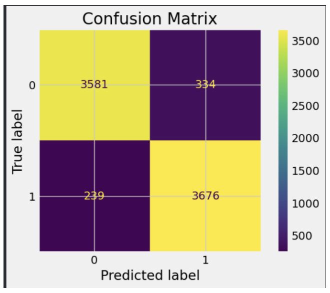


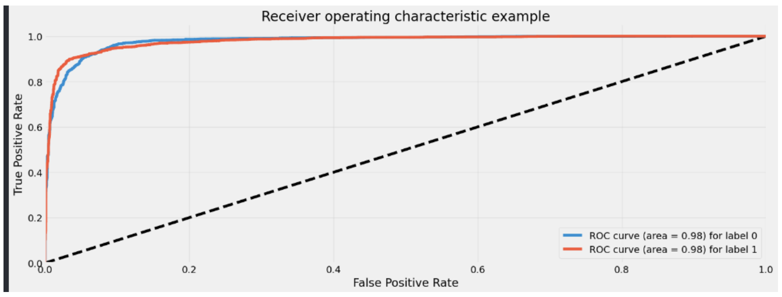
```
{'lgbm__bagging_fraction': 0.8705080567422835,
 'lgbm__bagging_freq': 4,
 'lgbm__feature_fraction': 0.5816499405911423,
 'lgbm__lambda_l1': 1.813533924696711e-07,
 'lgbm__lambda_l2': 1.2257456317807677e-06,
 'lgbm__min_child_samples': 85,
 'lgbm__num_leaves': 201}
```

Voting Classifier (LightGBM + XGBoost)

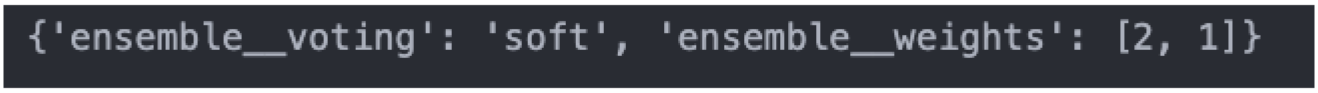


```
Model accuracy for train set: 1.000
Model accuracy for test set: 0.927

              precision    recall  f1-score   support

           0       0.94      0.91      0.93      3915
           1       0.92      0.94      0.93      3915

    accuracy                           0.93      7830
   macro avg       0.93      0.93      0.93      7830
weighted avg       0.93      0.93      0.93      7830
```

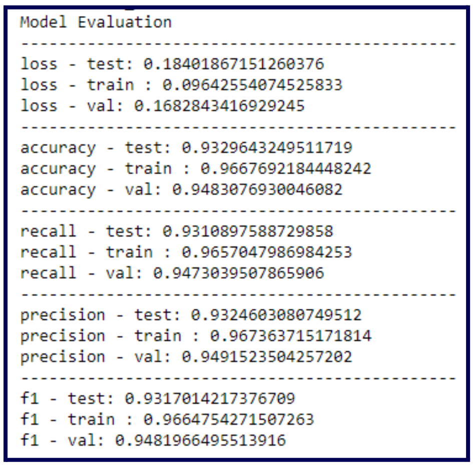
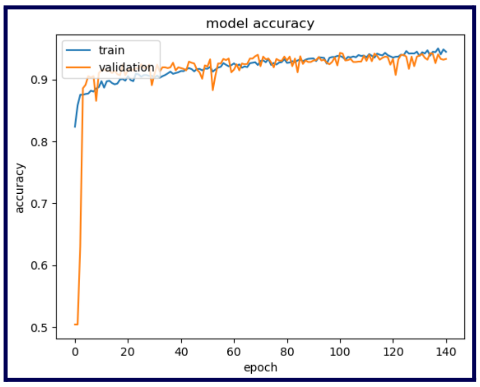
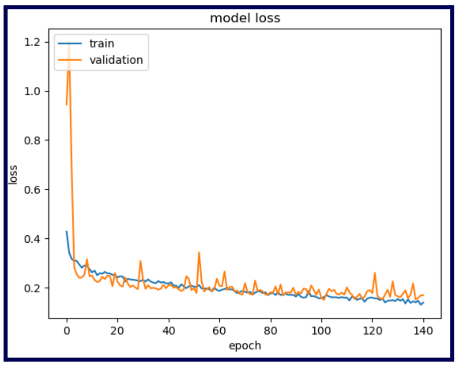




BEST PARAMETERS FOR Voting Classifier



```
{'ensemble__voting': 'soft', 'ensemble__weights': [2, 1]}
```

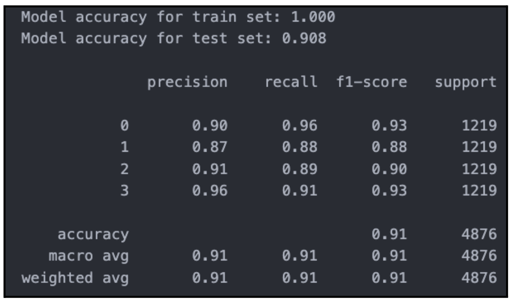
RCC_Net		 

```
Model Evaluation
---------------------------------------------
loss - test: 0.18401867151260376
loss - train : 0.09642554074525833
loss - val: 0.1682843416929245
---------------------------------------------
accuracy - test: 0.9329643249511719
accuracy - train : 0.9667692184448242
accuracy - val: 0.9483076930046082
---------------------------------------------
recall - test: 0.9310897588729858
recall - train : 0.9657047986984253
recall - val: 0.9473039507865906
---------------------------------------------
precision - test: 0.9324603080749512
precision - train : 0.967363715171814
precision - val: 0.9491523504257202
---------------------------------------------
f1 - test: 0.9317014217376709
f1 - train : 0.9664754271507263
f1 - val: 0.9481966495513916
```

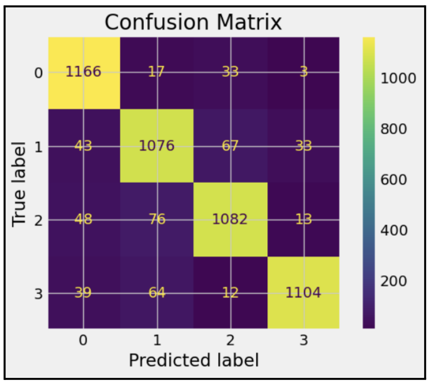
Table 12: Task 1 model results table

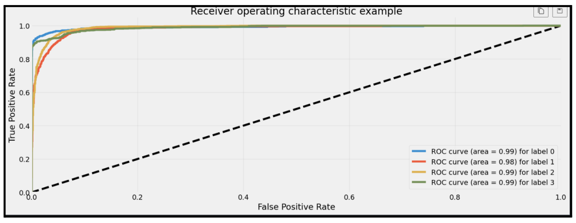
Task2 model result		
Model Name	**Score**	**Matrix & Curve**

XGBOOST_Standard

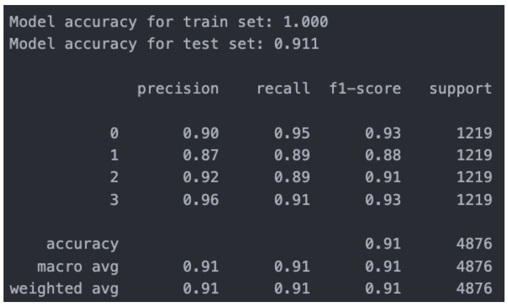


```
Model accuracy for train set: 1.000
Model accuracy for test set: 0.908

              precision    recall  f1-score   support

           0       0.90      0.96      0.93      1219
           1       0.87      0.88      0.88      1219
           2       0.91      0.89      0.90      1219
           3       0.96      0.91      0.93      1219

    accuracy                           0.91      4876
   macro avg       0.91      0.91      0.91      4876
weighted avg       0.91      0.91      0.91      4876
```

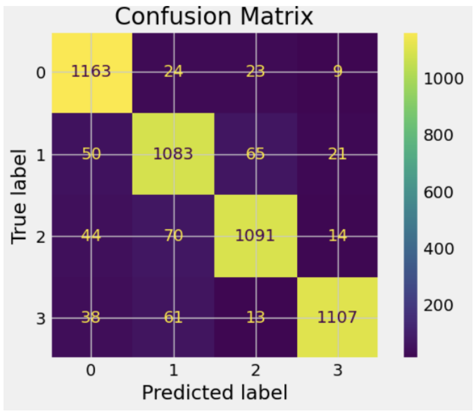


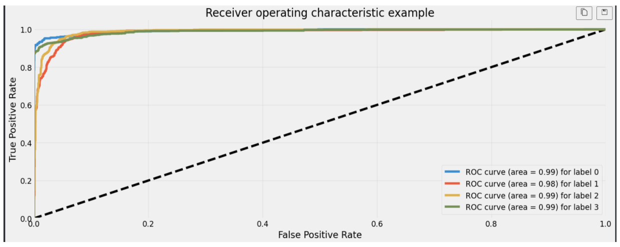


XGBoost Tuned

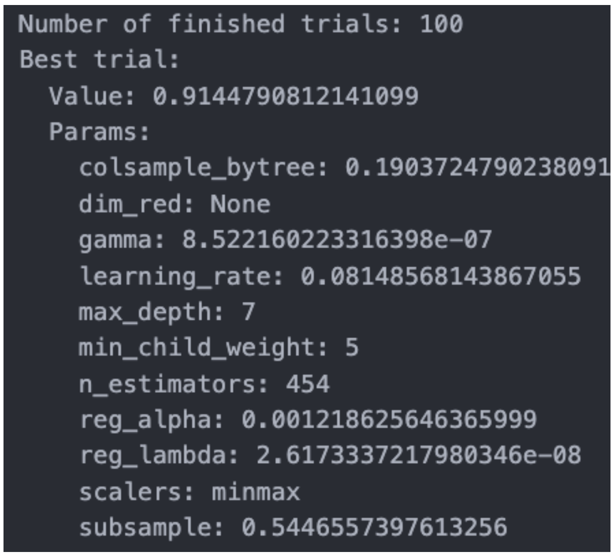


```
Model accuracy for train set: 1.000
Model accuracy for test set: 0.911

              precision    recall  f1-score   support

           0       0.90      0.95      0.93      1219
           1       0.87      0.89      0.88      1219
           2       0.92      0.89      0.91      1219
           3       0.96      0.91      0.93      1219

    accuracy                           0.91      4876
   macro avg       0.91      0.91      0.91      4876
weighted avg       0.91      0.91      0.91      4876
```



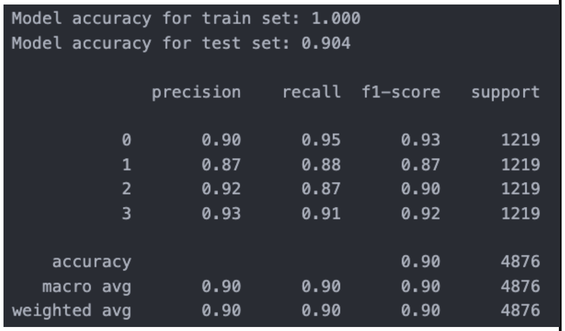


BEST PARAMETERS FOR XGBOOST

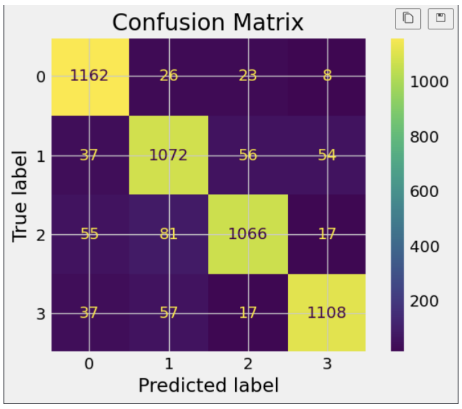


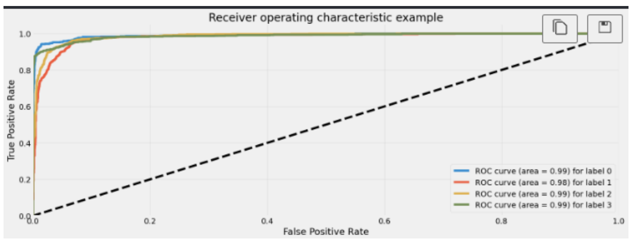
```
Number of finished trials: 100
Best trial:
  Value: 0.9144790812141099
  Params:
    colsample_bytree: 0.1903724790238091
    dim_red: None
    gamma: 8.522160223316398e-07
    learning_rate: 0.08148568143867055
    max_depth: 7
    min_child_weight: 5
    n_estimators: 454
    reg_alpha: 0.0012186256463659999
    reg_lambda: 2.6173337217980346e-08
    scalers: minmax
    subsample: 0.5446557397613256
```

LightGBM

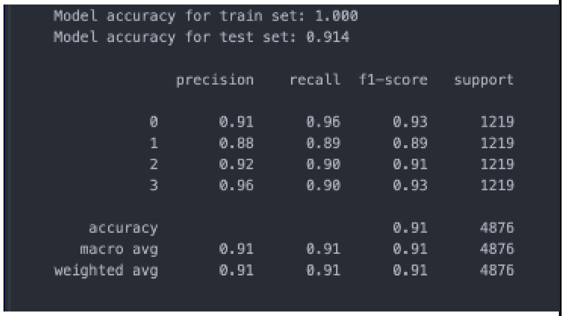


```
Model accuracy for train set: 1.000
Model accuracy for test set: 0.904

              precision    recall  f1-score   support

           0       0.90      0.95      0.93      1219
           1       0.87      0.88      0.87      1219
           2       0.92      0.87      0.90      1219
           3       0.93      0.91      0.92      1219

    accuracy                           0.90      4876
   macro avg       0.90      0.90      0.90      4876
weighted avg       0.90      0.90      0.90      4876
```

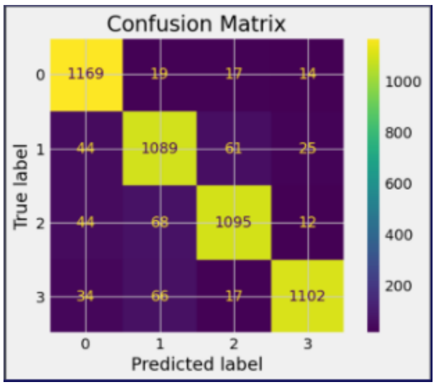


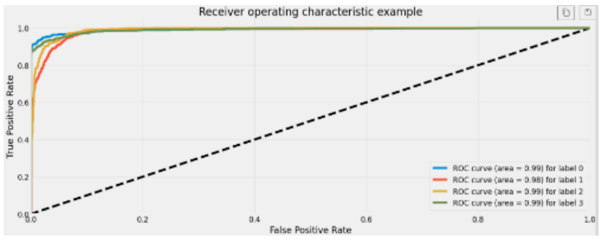


LightGBM Tuned

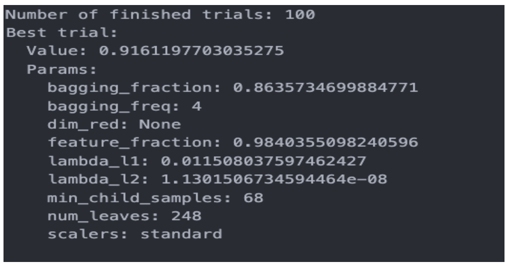


```
Model accuracy for train set: 1.000
Model accuracy for test set: 0.914

              precision    recall  f1-score   support

           0       0.91      0.96      0.93      1219
           1       0.88      0.89      0.89      1219
           2       0.92      0.90      0.91      1219
           3       0.96      0.90      0.93      1219

    accuracy                           0.91      4876
   macro avg       0.91      0.91      0.91      4876
weighted avg       0.91      0.91      0.91      4876
```

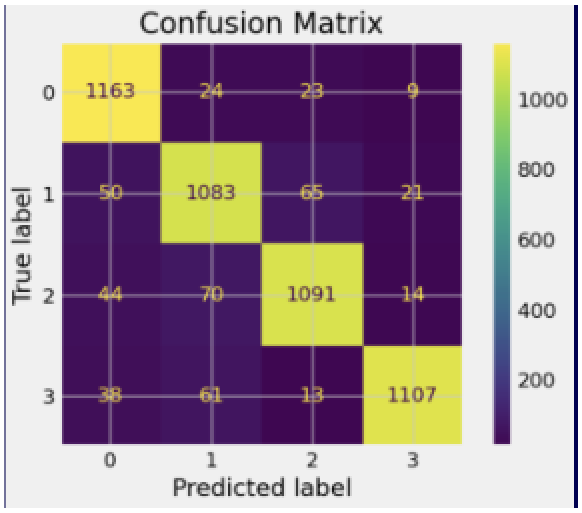
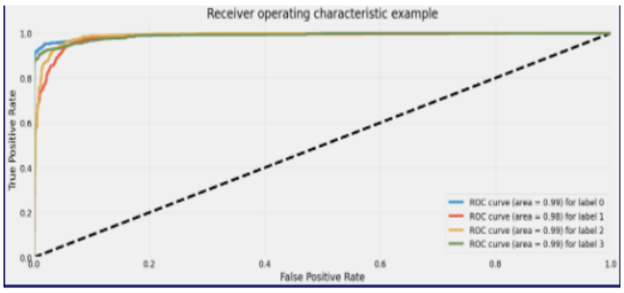


BEST PARAMETERS FOR LIGHTGBM



```
Number of finished trials: 100
Best trial:
  Value: 0.9161197703035275
  Params:
    bagging_fraction: 0.8635734699884771
    bagging_freq: 4
    dim_red: None
    feature_fraction: 0.9840355098240596
    lambda_l1: 0.011508037597462427
    lambda_l2: 1.1301506734594464e-08
    min_child_samples: 68
    num_leaves: 248
    scalers: standard
```

Voting Classifier (LightGBM + XGBoost)		 

BEST PARAMETERS FOR Voting Classifier

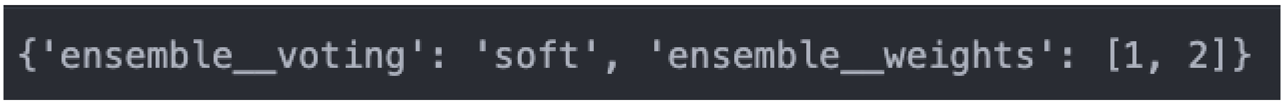
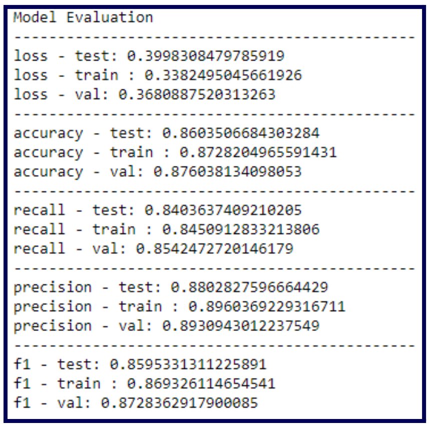
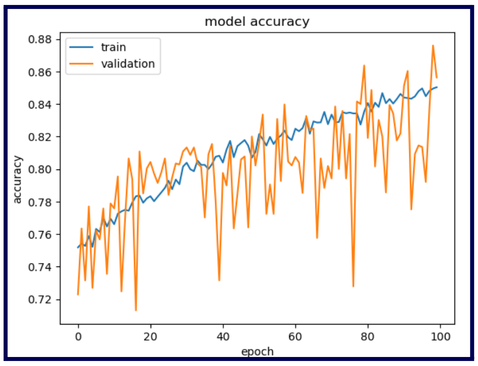
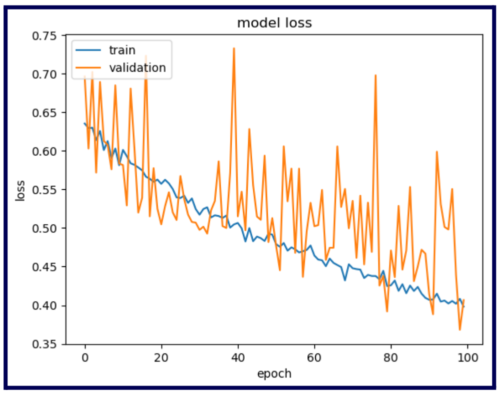
	 `{'ensemble__voting': 'soft', 'ensemble__weights': [1, 2]}`	
RCCnet	 Model Evaluation -- loss - test: 0.3998308479785919 loss - train : 0.3382495045661926 loss - val: 0.3680887520313263 -- accuracy - test: 0.8603506684303284 accuracy - train : 0.8728204965591431 accuracy - val: 0.876038134098053 -- recall - test: 0.8403637409210205 recall - train : 0.8450912833213806 recall - val: 0.8542472720146179 -- precision - test: 0.8802827596664429 precision - train : 0.8960369229316711 precision - val: 0.8930943012237549 -- f1 - test: 0.8595331311225891 f1 - train : 0.869326114654541 f1 - val: 0.8728362917900085 **Fig: Tuned RCC_Net accuracy score**	 **Fig :Tuned RCC_Net model accuracy**  **Fig :Tuned RCC_Net model loss**

Table 13: Task 2 model results table