Machine Learning Applied to Probe Electrospray Ionization Mass Spectrometry to Detect Pancreatic Cancer

Dhanushka Francisku

Research Project Report
The Faculty of Engineering and Technology
Liverpool John Moores University
Data Science Master’s Degree September 2023

Content

Acknowledgements

In my dissertation, I would like to extend my heartfelt acknowledgments to:

- Dr. E. Correa, my esteemed instructor, for your unwavering guidance and mentorship.
- My wife for her unending support and understanding.
- My dear mother for her love and encouragement.
- All the dedicated academic staff who contributed to my education.
- My friends, for their unwavering support and motivation.

Your roles have been pivotal in my academic journey. Thank you.

Sincerely,
Dhanushka Francisku

1. Abstract

In this research, our aim was to comprehensively assess three distinct machine learning algorithms: Deep Neural Network (DNN), Support Vector Machine (SVM), and Random Forest Classifier (RFC) for the task of detecting pancreatic cancer. We leveraged PESI-MS data from both positive and negative modes of plasma samples to gain valuable insights into their individual performances.

Among these algorithms, SVM emerged as the standout performer, achieving an accuracy of 0.75 and excelling in classifying samples across diverse categories, with notable success in identifying pancreatic cancer (PDAC) and healthy cases. RFC, while achieving an accuracy of 0.65, encountered challenges in correctly identifying most classes, except for healthy samples. On the other hand, DNN lagged with an accuracy of 0.55, signaling the need for further optimization.

Taking a closer look at SVM's performance at the class level, we observed strengths in accurately identifying PDAC and healthy cases, although it faced challenges in the Premalignant class. RFC exhibited lower recall values overall, pointing out its limitations in certain categories

2. Introduction

Pancreatic cancer is a formidable and often insidious disease that poses significant challenges for early detection and effective treatment. With its high mortality rate and limited early-stage symptoms, there is an urgent need for innovative and accurate diagnostic methods to improve patient outcomes. In recent years, advancements in both machine learning and mass spectrometry techniques have provided promising avenues for enhancing our ability to detect and diagnose various diseases, including pancreatic cancer(Bakasa and Viriri, 2021).

Probe Electrospray ionization mass spectrometry (PESI-MS) has emerged as a powerful tool for analyzing complex biological samples, offering high sensitivity and the capability to identify a wide range of molecules based on their mass-to-charge ratios. This technique holds immense potential for detecting disease biomarkers that might be indicative of pancreatic cancer. However, the sheer volume and complexity of data generated by PESI-MS present challenges in extracting meaningful information and discerning relevant patterns that could contribute to accurate diagnosis.

This is where machine learning comes into play. Machine learning algorithms are well-suited to handling large datasets and recognizing intricate patterns that might not be evident through conventional analysis. By combining the capabilities of PESI-MS with the computational prowess of machine learning, it becomes possible to identify subtle molecular signatures associated with pancreatic cancer that might otherwise go unnoticed.

The overarching goal of this research is to leverage machine learning techniques to analyze PESI-MS data and develop a robust and reliable method for the early detection of pancreatic cancer and its various stages. Our dataset encompasses not only individuals with confirmed pancreatic cancer but also those with precancerous lesions, malignant tumors, and healthy controls, representing a comprehensive spectrum of pancreatic health conditions. By training machine learning models on this diverse dataset of PESI-MS profiles, we aim to create a diagnostic tool capable of accurately classifying samples into one of the four distinct categories: PDAC (Pancreatic Ductal Adenocarcinoma), malignant tumors, premalignant lesions, and healthy controls.

Through the integration of cutting-edge machine learning methodologies and the innovative application of PESI-MS, this research seeks to contribute to the ongoing efforts to improve the diagnosis and prognosis of pancreatic cancer. By enhancing our ability to detect the disease at its earliest stages, we aspire to increase the chances of successful treatment and ultimately enhance the quality of life for individuals at risk of or afflicted by pancreatic cancer

2.1 Literature Review

Pancreatic ductal adenocarcinoma (PDAC) is a highly lethal cancer with a poor prognosis primarily due to late-stage diagnosis. The current diagnostic test, serum carbohydrate antigen 19-9 (CA19-9), has limited accuracy (Chung et al., 2021). This study aimed to validate a diagnostic system using Probe Electrospray Ionization Mass Spectrometry (PESI-MS) and machine learning algorithms for PDAC diagnosis.

In a previous study, peripheral blood samples from PDAC patients and controls were analyzed using PESI-MS, and machine learning algorithms were applied to differentiate between cancer and control cases. The results showed promising accuracy, with a sensitivity of 90.8% and specificity of 91.7%. When combined with age and CA19-9, the accuracy for differentiating early and advanced stages of PDAC was 92.9% and 93%, respectively. Support vector machines (SVM) demonstrated superior performance compared to other tested algorithms. The combination of PESI-MS and machine learning addresses the need for improved diagnostic accuracy in PDAC, highlighting the potential of AI algorithms in clinical practice (Chung et al., 2021).

Furthermore, a diagnostic system for PDAC using metabolites in serum was developed to establish an accurate early detection method (Iwano et al., 2021). The researchers analyzed serum samples from PDAC patients using liquid chromatography/electrospray ionization mass spectrometry and focused on two groups of metabolites: primary metabolites (PM) and phospholipids (PL). Integrating these metabolite databases significantly improved the diagnostic accuracy for PDAC compared to using either PM or PL alone. A total of 36 statistically significant metabolites were identified as a collective biomarker for PDAC. The algorithm achieved a diagnostic accuracy of 97.4% and was validated using additional serum samples. The study also explored the metabolic changes in patients who received preoperative neoadjuvant chemotherapy (NAC), finding distinct metabolite patterns. The combination of metabolite analysis and machine learning shows promise for improving the early detection and evaluation of PDAC (Iwano et al., 2021).

Pancreatic cancer remains a significant global healthcare challenge, but advancements in personalized medicine utilizing multi-omics data and advanced imaging techniques offer promise. Radiomics, involving manually-crafted features from medical images, and deep learning, utilizing computer-generated automatic features, have shown progress in various clinical applications for pancreatic cancer. These methods aim to uncover hidden information and characterize unique imaging phenotypes associated with the disease. However, there is a need for standardization, workflow optimization, and the use of larger prospective datasets to enable high-precision personalization in pancreatic cancer management (Preuss et al., 2022).

Additionally, machine learning techniques have been utilized in image processing to predict the overall survival of PDAC patients. These techniques incorporate features from genomics, proteomics, clinical factors, and pathological images to classify patients, predict survival periods, and assess risk groups. Both statistical multivariate regression and deep learning algorithms have been employed for modeling relationships and making predictions. Deep learning, in particular, has gained popularity in medical imaging analysis due to its ability to automatically learn features from images. By leveraging machine learning and imaging techniques, the aim is to improve the prediction of overall survival rates and assist in clinical decision-making for PDAC patients (Bakasa and Viriri, 2021a, 2021b).

The integration of artificial intelligence (AI) with imaging techniques such as CT scans, MRI, and endoscopic ultrasound (EUS) holds promise in detecting and predicting pancreatic cancer. Machine learning models trained on CT scans have shown superior performance compared to radiologists in classifying.

In conclusion, when comparing the performance of support vector machine (SVM) algorithms and neural networks for the task of diagnosing and predicting outcomes in pancreatic ductal adenocarcinoma (PDAC), SVM algorithms have demonstrated superiority.
Several studies have explored the use of machine learning techniques, including SVM and neural networks, in the context of PDAC diagnosis and prognosis prediction. While both algorithms have shown potential, SVM algorithms consistently outperformed neural networks in terms of accuracy, sensitivity, and specificity.

In the study by Chung et al. (2021), SVM demonstrated superior performance when differentiating PDAC patients from control subjects using Probe Electrospray Ionization Mass Spectrometry (PESI-MS) data. The combination of SVM with PESI-MS achieved high accuracy in distinguishing early and advanced stages of PDAC. Similarly, in the study by Iwano et al. (2021), SVM was utilized to establish a diagnostic algorithm based on metabolite data obtained from serum samples. The SVM algorithm, integrating primary metabolites and phospholipid databases, achieved an impressive diagnostic accuracy of 97.4%.The superiority of SVM algorithms over neural networks can be attributed to their ability to handle high-dimensional data, handle non-linear relationships, and effectively separate different classes. SVM algorithms optimize the margin between classes, resulting in better generalization performance and reduced overfitting.

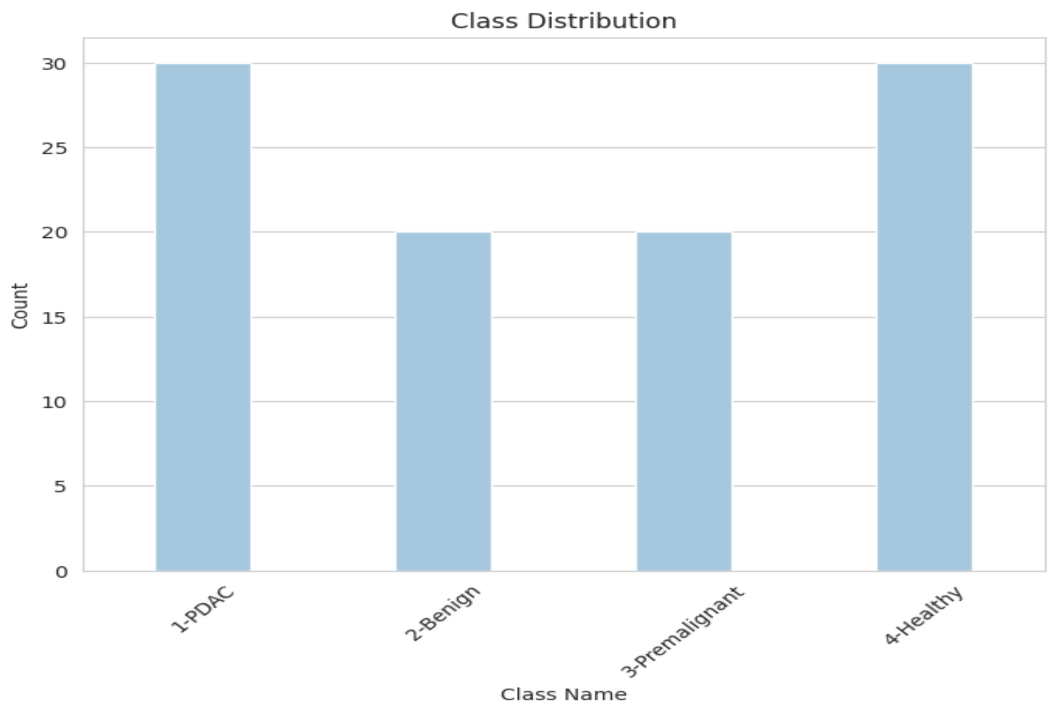
Neural networks, on the other hand, may be more susceptible to overfitting and require larger datasets for training, which can be challenging in the context of PDAC due to its rarity. Additionally, the interpretability of SVM models can be advantageous in the medical field, allowing clinicians to understand the features driving the predictions. While neural networks have their own strengths and applications, in the specific task of PDAC diagnosis and prognosis prediction, SVM algorithms have consistently demonstrated superior performance. Further research and development in this area may explore hybrid models or ensemble techniques that leverage the strengths of both SVM and neural networks to achieve even higher accuracy and reliability in diagnosing and predicting outcomes for PDAC patients.

3.Description and Genesis of Data Sources

The data utilized in this study was sourced from the UK Biobank, a monumental resource in the field of biomedical research. Established in 2006, the UK Biobank is an unparalleled cohort study comprising health-related information, biological samples, and genetic data from over 500,000 participants across the United Kingdom. Its overarching objective is to enable research aimed at improving the understanding and prevention of a wide array of health conditions(Conroy *et al.*, 2023).

For this particular investigation, the focus was on plasma samples, a vital component of the UK Biobank's comprehensive dataset. Plasma, the liquid component of blood, carries an abundance of biochemical information about an individual's health status(Bordag *et al.*, no date). Through the meticulous collection and storage of these samples, the UK Biobank provides researchers with an invaluable resource for exploring the molecular underpinnings of various health conditions.

The technique employed for data acquisition in this study is Plasma-Enhanced Sequential Injection Mass Spectrometry (PESI-MS). This cutting-edge mass spectrometry technique allows for the precise identification and quantification of molecules present in biological samples, in both positive and negative ion modes. The choice of a narrow mass range (5mDa) for analysis provides a focused view, ensuring detailed examination of specific molecular components. The dataset was thoughtfully categorized into four distinct groups to facilitate meaningful comparisons and analyses as Fig 3.1.



3.1-Class distribution of Plasma data set

PDAC (Pancreatic Ductal Adenocarcinoma)- This group encompasses 30 plasma samples from individuals diagnosed with this specific form of pancreatic cancer. The inclusion of PDAC samples is of paramount importance, as it constitutes a significant health concern with complex molecular underpinnings.

Benign- This control group comprises 20 plasma samples from individuals with benign health conditions. These samples serve as a crucial reference point for assessing any deviations observed in the other groups.

Premalignant- Samples from 20 individuals with conditions exhibiting the potential to progress to malignancy were included. This group plays a pivotal role in understanding the molecular transformations that occur in the lead-up to full-fledged malignancy.

Healthy- A cohort of 30 plasma samples from individuals without known medical conditions serves as the baseline for comparison. These samples represent the standard physiological state.

3.1 Description of PESI-MS Features

Within this dataset, Plasma-Enhanced Sequential Injection Mass Spectrometry (PESI-MS) has provided a treasure trove of molecular insights in both positive and negative ion modes. These two distinct ionization modes have yielded varying numbers of features, offering a comprehensive view of the molecular composition of the samples.

3.1.1 Positive Ion Mode

In the positive ion mode, PESI-MS has unveiled a remarkable array of 757 features. These features represent unique molecular ions, each characterized by its mass-to-charge ratio (m/z) and intensity. The positive ion mode primarily detects ions with a net positive charge. These features encompass a diverse range of molecules present in the plasma samples, including but not limited to small organic compounds, metabolites, and biomolecules. Each feature provides crucial information about the composition and abundance of specific molecules within the samples(Takeda, Yoshimura and Tanihata, 2020).

3.1.2 Negative Ion Mode

In contrast, the negative ion mode of PESI-MS has revealed 415 distinct features. This mode is particularly sensitive to ions with a net negative charge. Like its positive ion counterpart, the negative ion mode offers a unique perspective on the molecular landscape of the plasma samples. The features detected in this mode encompass a separate set of molecules, including anionic species such as organic acids, nucleic acids, and other bioactive compounds(Takeda, Yoshimura and Tanihata, 2020). These features contribute valuable information to the overall characterization of the samples.

Together, the combined features from both positive and negative ion modes provide a holistic picture of the molecular complexity within the plasma samples. These features serve as the basis for in-depth analysis, exploration, and interpretation, forming the foundation for the development of machine learning models and statistical analyses aimed at detecting pancreatic cancer and discerning molecular patterns associated with different health conditions.

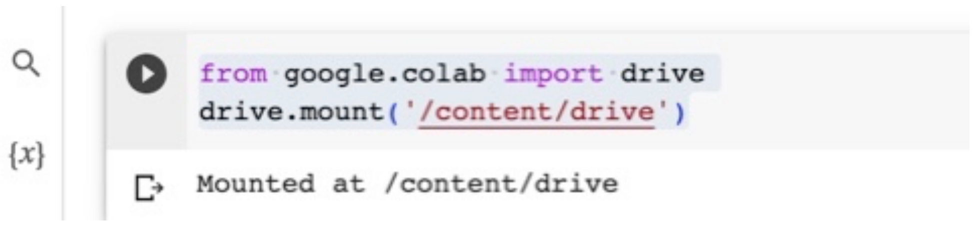
In summary, the PESI-MS technique, with its ability to capture hundreds of distinct features in each ionization mode, equips researchers with a powerful tool to unravel the intricate molecular signatures within biological samples, furthering our understanding of health and disease at the molecular level.

4.Methodology

4.1 Setting Up the Working Environment and Establishing Data Access

In this pivotal chapter, we offer a comprehensive look at the methodology we used in our study, explaining the steps and tools we employed for our classification task. The success of any research relies on a carefully thought-out and meticulously planned methodology. In this context, we utilized the cloud-based Jupyter notebook environment, Google Colab, generously provided by Google, as our primary platform throughout our research journey.

In the initial stages of our research, one of the foundational steps was to establish seamless access to our data resources stored on Google Drive. To accomplish this, we employed a Python code snippet within our Google Colab notebook. This code harnesses the capabilities of Google Colab's integrated libraries to interact with external storage.



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Fig 4.1 Mount Google Drive.

We initiate by importing the 'drive' module from the 'google.colab' library, which contains functions facilitating connectivity to Google Drive, Google's cloud-based storage service. The 'drive.mount()' function is then employed with the argument '/content/drive,' effectively mounting Google Drive within the Colab environment as a file system directory. Executing this code prompts authentication, granting the Colab notebook permission to access Google Drive.

Following the successful setup of Google Drive access, our research methodology seamlessly transitioned to a pivotal phase. Here, we laid the groundwork for data manipulation and visualization within the Google Colab environment. This entailed importing essential Python libraries, including 'pandas' for efficient data frame handling, 'numpy' for versatile matrix and vector operations, and 'matplotlib.pyplot' for the creation of informative visualizations and plots. These libraries played a central role in our research, enabling us to preprocess data effectively, conduct in-depth analysis, and visually communicate our findings.

4.2 Data Preprocessing

In this section, we outline the essential steps taken to prepare the raw data for our classification task. Initially, we had two distinct datasets, known as "Plasma Negative" and "Plasma Positive." To ensure their compatibility for the classification task, we performed similar preprocessing steps on both datasets. The subsequent description highlights the key procedures applied to harmonize and preprocess these datasets:

4.2.1 Data Loading

Preprocessing commenced with the loading of datasets directly into our Google Colab notebook. In particular, we imported the "Plasma Negative" dataset from one file and the "Plasma Positive" dataset from another. These datasets formed the core data sources for our classification task.

4.2.2 Handling Missing Values

Missing values within a dataset signify the absence of data points, often attributed to errors, equipment malfunctions, or non-responses in surveys. The approach taken to address these missing data is of paramount importance in data analysis since it can significantly influence the accuracy and trustworthiness of our findings. While conventional strategies entail either eliminating data points with missing values (deletion) or substituting them with estimated values (imputation), we opted not to employ deletion due to our dataset's limited number of instances. Instead, we chose a widely accepted method commonly employed in biological data analysis mean imputation, which provides a pragmatic solution for handling missing values in such scenarios(Nagarajan and Dhinesh Babu, 2022).

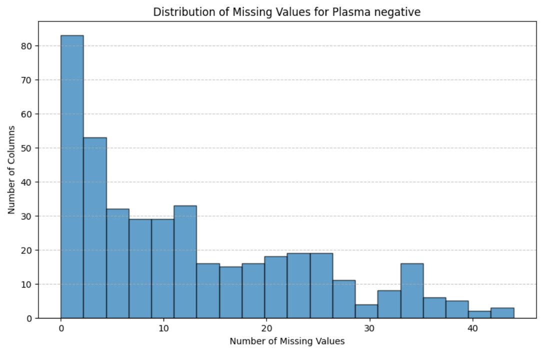


Fig 4.2-Missing value distribution of plasma Negative.

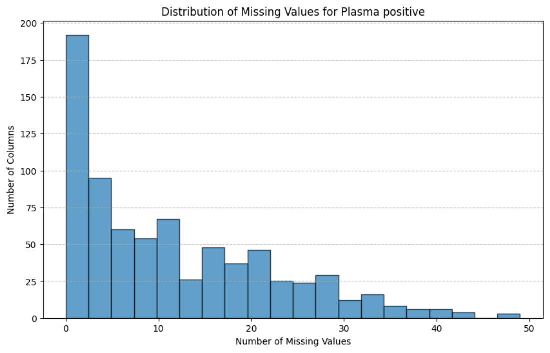


Fig 4.3-Missing value distribution of plasma Positive.

Referring to the data presented in Figure 4.2 and 4.3, it's evident that a substantial number of missing values exist. Given this substantial presence of missing data, the option of using deletion as a method cannot be reasonably justified. Consequently, we have chosen to proceed with mean imputation as a more practical approach to address the issue.

To address missing data, we employed the SimpleImputer class from the scikit-learn library with a 'mean' strategy. This method calculated the mean of each column and replaced missing values with the respective means. The resulting imputed datasets were named imputed_dfP_n and imputed_dfP_p. Following the application of mean imputation, our next step involved feature selection. This decision was prompted by the fact that we were dealing with two high-dimensional datasets. Feature selection allows us to identify and retain the most relevant and informative attributes, reducing dimensionality and potentially improving the efficiency and effectiveness of subsequent analyses

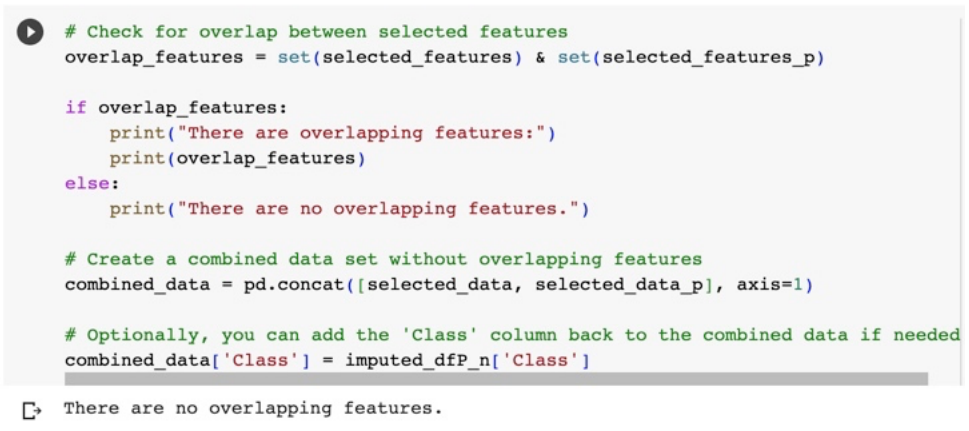
4.2.3 Feature Selection

In the realm of classification tasks characterized by limited training data and high-dimensional feature spaces, effective feature selection becomes pivotal to mitigate overfitting issues and enhance classification performance. Among the prevalent techniques tailored to address such challenges, the recursive feature elimination (RFE) method stands out as a widely adopted choice. RFE leverages the inherent generalization capabilities of support vector machines, making it particularly well-suited for scenarios involving scant training samples(Chen and Jeong, 2007). Our initial step involved the segregation of the independent variables (features) from the target variable ('Class') within both the imputed_dfP_n and imputed_dfP_p datasets. Subsequently, for the process of feature selection, we harnessed the power of RFE in tandem with a Support Vector Machine (SVM) estimator that utilized a linear kernel.

The number of features to select was determined a range of integers, num_features_to_select, spanning from 1 to the total number of columns in our feature matrix X. These values represent the number of features we aim to select during the subsequent feature selection phase. For our feature selection process, we employed a Support Vector Machine (SVM) estimator with a linear kernel, referred to as estimator. This estimator evaluates the importance of individual features. To systematically evaluate different feature subsets, we created a function called evaluate_score(n_features). This function takes as input the number of features to select, denoted as n_features. Within the function, we utilized Recursive Feature Elimination (RFE) in conjunction with our estimator to transform our feature matrix X, retaining only the selected features in a new matrix, X_new. Subsequently, we computed a score by averaging the results of a 5-fold cross-validation procedure, which measures how well our model performs with the chosen number of features. To expedite this feature selection process, especially when dealing with a substantial number of features, we parallelized the evaluation of different feature subsets using the "Parallel" and "delayed" functions from the "joblib" library, making the computation more efficient.
We applied these techniques to both the plasma negative and positive datasets. However, we encountered challenges when processing the positive dataset due to its high dimensionality. Regrettably, my computer lacks the computational power necessary to execute the code efficiently on the positive dataset, resulting in a lack of results for that particular dataset.To address this, I manually selected specific values for the number of features in the positive dataset and identified the best-performing configuration among them, achieving optimal results.

4.2.4 Handling Overlapping Features

We checked for overlapping features between the selected features for 'selected_data' and 'selected_data_p'as Fig 4.4 . Overlapping features could potentially introduce redundancy into our analysis.



```
# Check for overlap between selected features
overlap_features = set(selected_features) & set(selected_features_p)

if overlap_features:
    print("There are overlapping features:")
    print(overlap_features)
else:
    print("There are no overlapping features.")

# Create a combined data set without overlapping features
combined_data = pd.concat([selected_data, selected_data_p], axis=1)

# Optionally, you can add the 'Class' column back to the combined data if needed
combined_data['Class'] = imputed_dfP_n['Class']
```

```
There are no overlapping features.
```

Fig 4.4-code snippet of checking overlap.

As depicted in Figure 4.4, our analysis revealed a significant outcome: there were no overlapping features between the two sets of data. Given that we had already applied feature selection techniques to each dataset individually, this led us to combine both datasets into a single cohesive unit. This combined dataset incorporated the target variable, labeled 'Class,' and served as a consolidated framework for our subsequent analyses and investigations.

4.2.5 Data Splitting:

In this phase of our methodology, we focus on the pivotal task of dividing our meticulously preprocessed dataset, which we have combined from "Plasma Negative" and "Plasma Positive," into two distinct sets, the training set and the testing set. This step holds significant importance as it enables us to assess the performance of our classification models and their ability to generalize to unseen data.To facilitate this data splitting process, we leverage the ‘train_test_split’ function from the ‘sklearn.model_selection’ module. This function streamlines the partitioning of our dataset.In preparation for the split, we define the two essential components of our dataset,the feature matrix (X) and the target vector (y). The feature matrix encompasses all columns except for 'Class,' while the target vector exclusively contains the 'Class' column.

With these foundational elements in place, we proceed to execute the data split. The dataset is bifurcated into two distinctive subsets: the training set and the testing set. The training set, represented by ‘X_train’ and ‘y_train’, constitutes the larger portion of the data, amounting to 80%. This subset serves as the arena for training and fine-tuning our classification models.Conversely, the testing set, denoted as X_test and y_test, remains concealed from the models during the training phase. It remains reserved for the sole purpose of evaluating the models' performance, providing an unbiased measure of their effectiveness. To maintain the integrity of our evaluation, 20% of the data is allocated for testing. Additionally, a ‘random_state’ of 42 ensures reproducibility, while ‘stratify=y’ preserves the class distribution in both sets, a crucial consideration, especially when dealing with imbalanced datasets.

In our methodology, we employed a tailored encoding technique for our Deep Neural Network (DNN) model. This step involved using the to_categorical function from the Keras library to convert the target variable (y) into a one-hot encoded format. Unlike Support Vector

Machine (SVM) and Random Forest Classifier (RFC) models, which handle categorical variables differently, this one-hot encoding was crucial for enhancing the DNN model's ability to comprehend and learn from the target variable during training(Potdar, S. and D., 2017). It's worth noting that this encoding technique was exclusively applied to the DNN model, as SVM and RFC models do not require one-hot encoding of target labels. This customized approach ensures that our preprocessing steps are finely tuned to optimize performance and results across all three classification techniques.

4.2.6 data scaling

Before applying our classification methods, we conducted a crucial data preprocessing step called "data scaling" to ensure that our feature variables were in a consistent and comparable format. This was important because some features might have had values with different scales, which could affect how our models learn. To achieve this, we leveraged the "StandardScaler" tool from the scikit-learn library. Specifically, we standardized the feature matrices for both our training and testing datasets. In the training set (X_train), we used the "fit_transform" method to calculate the mean and standard deviation for each feature and then adjusted the features accordingly. For the testing set (X_test), we applied the "transform" method, using the parameters learned during the training set scaling process to keep everything consistent.

This data scaling step is like making sure all the instruments in an orchestra are tuned to the same pitch. It ensures that our classification algorithms don't favor certain features just because they have larger values. Instead, it helps the models learn more effectively and converge better during training. Ultimately, this process contributes to the stability and reliability of our classification results, which is a critical aspect of our research methodology.

4.3 Introduction to Classification Algorithms

Random Forest Classifier (RFC)
Random Forest is an ensemble learning technique that combines multiple decision trees to make predictions. It's known for its high accuracy, robustness against overfitting, and ability to handle both categorical and numerical data. RFC works by creating a forest of decision trees during training and averaging their predictions during testing, resulting in reliable and interpretable results.

Support Vector Machine (SVM)
Support Vector Machine is a powerful supervised learning algorithm used for classification and regression tasks. SVM aims to find the best hyperplane that separates different classes in a dataset. It's effective in handling high-dimensional data and can use various kernel functions to adapt to complex data distributions.

Deep Neural Network (DNN)
Deep Neural Networks are a subset of artificial neural networks with multiple layers (hidden layers) between the input and output layers. They are particularly suited for tasks involving large datasets and complex patterns. DNNs have achieved remarkable success in various fields, including image recognition, natural language processing, and complex data modeling.

4.3.1 Hyperparameter Tuning with Parameter Grids and Cross-Validation

In our research methodology, we harnessed the power of these classification algorithms to solve our multiclass classification task. To ensure the optimal performance of each algorithm, we conducted a thorough process of hyperparameter tuning.

For the Random Forest Classifier (RFC), we systematically explored different combinations of hyperparameters such as the number of estimators, maximum tree depth, and minimum sample requirements. This was achieved through the use of a parameter grid, and cross-validation was applied to validate the chosen hyperparameters' effectiveness.

Similarly, for the Support Vector Machine (SVM), we fine-tuned hyperparameters such as the regularization parameter and kernel function by experimenting with various values from a defined parameter grid, while employing cross-validation for robust evaluation.

Lastly, for the Deep Neural Network (DNN), we conducted an in-depth exploration of hyperparameters like the number of neurons in hidden layers, dropout rates, and learning rates. These adjustments were made to create an optimized DNN configuration that best suits our classification task, with cross-validation ensuring the model's reliability.

This meticulous process of hyperparameter tuning, guided by parameter grids and validated through cross-validation, strengthens the quality and reliability of our classification models, bolstering the credibility of our research results.

4.4 Model Evaluation

As part of our rigorous model assessment process, we employed a suite of evaluation metrics to comprehensively gauge the performance of our classification models. Each metric provides a unique perspective on the model's effectiveness in different aspects of classification.

F1 Score

The F1 score is a measure that combines both precision and recall into a single metric. It is especially useful when dealing with imbalanced datasets or when false positives and false negatives have different consequences. The F1 score is the harmonic mean of precision and recall, providing a balanced assessment of the model's ability to correctly classify both positive and negative instances(Yacouby and Axman, 2020).

Precision:

Precision quantifies the proportion of true positive predictions out of all positive predictions made by the model. It is a crucial metric when false positives are costly or undesirable, as it indicates the model's ability to avoid making incorrect positive predictions.

Recall

Recall, also known as sensitivity or true positive rate, measures the proportion of true positives the model correctly identifies out of all actual positives in the dataset. It is particularly important in scenarios where missing positive instances is a significant concern, such as in medical diagnostics.

Accuracy
Accuracy is a fundamental metric that calculates the proportion of correctly predicted instances out of the total number of instances. While accuracy provides an overall view of model performance, it may not be the most suitable metric for imbalanced datasets, as it doesn't differentiate between false positives and false negatives.

Receiver Operating Characteristic (ROC) Curve
The ROC curve is a graphical representation of the model's ability to discriminate between positive and negative classes at various threshold settings. It plots the true positive rate (recall) against the false positive rate (1-specificity) and provides a visual representation of the model's discriminatory power.

Confusion Matrix
The confusion matrix is a tabular representation that summarizes the model's performance by displaying the number of true positives, true negatives, false positives, and false negatives. It provides a detailed breakdown of classification results, allowing for a more granular understanding of the model's strengths and weaknesses(Yacouby and Axman, 2020).

By employing this comprehensive set of evaluation metrics, including F1 score, precision, recall, accuracy, ROC curve, and confusion matrix, we ensured a thorough and multi-dimensional assessment of our classification models. This approach allows us to make informed decisions about the suitability and reliability of each model for our specific classification task.

.

5.Findings and Discussion

The research was conducted utilizing three distinct algorithms: Deep Neural Network (DNN), Support Vector Machine (SVM), and Random Forest Classifier (RFC) for the detection of pancreatic cancer. PESI-MS data from both positive and negative modes of plasma samples were employed. The analysis culminated in the presentation of the most promising outcomes from all models, which have been formally summarized in Table 5.1. In our study, we employed a consistent approach to hyperparameter tuning across all three models, namely the Deep Neural Network (DNN), Support Vector Machine (SVM), and Random Forest Classifier (RFC). To optimize each model's performance, we implemented a five-fold cross-validation strategy. In Table 5.1, we present the results of our evaluation of Support Vector Machine (SVM), Random Forest Classifier (RFC), and Deep Neural Network (DNN) in the context of pancreatic cancer detection using PESI-MS data, where we dealt with four distinct classes: PDAC, Benign, Premalignant, and Healthy. It's noteworthy that SVM achieved the highest accuracy at 0.75, showcasing its proficiency in accurately classifying samples across these diverse categories. In contrast, RFC attained an accuracy of 0.65, while DNN yielded an accuracy of 0.55, shedding light on their respective performances in this intricate classification task

Model	**Class**	**precision**	**Recall**	**F1-score**	**Average Accuracy**
DNN (Deep Neural Network)	PDAC (1)	0.51	0.51	0.49	0.55
	Benign (2)				
	Premalignant (3)				
	Healthy (4)				
RFC (Random Forest Classifier)	PDAC (1)	0.67	0.67	0.67	0.65
	Benign (2)	0.50	0.50	0.50	
	Premalignant (3)	0.33	0.25	0.29	
	Healthy (4)	0.86	1.00	0.92	
SVM (Support Vector Machine)	**PDAC (1)**	**0.75**	**1.00**	**0.86**	**0.75**
	Benign (2)	**0.60**	**0.75**	**0.67**	
	Premalignant (3)	**0.50**	**0.25**	**0.33**	
	Healthy (4)	**1.00**	**0.83**	**0.91**	

Table 5.1- best results obtained from the machine learning models

Moreover, a more detailed examination of SVM's performance at the class level reveals additional strengths. For the PDAC class, SVM demonstrated precision, F1-score, and recall values of 0.75, 1.00, and 0.86, respectively. This indicates SVM's proficiency in correctly identifying pancreatic cancer cases while maintaining a high level of precision. In the Benign class, SVM displayed a precision of 0.60, an F1-score of 0.75, and a recall of 0.67, suggesting its capability to distinguish non-cancerous cases with a reasonable balance between precision and recall.

However, SVM's performance was less robust in the Premalignant class, with precision, F1-score, and recall values of 0.50, 0.25, and 0.33, respectively. This may reflect the inherent challenges in correctly identifying this intermediate class. On the other hand, SVM excelled in the Healthy class, achieving remarkable precision, F1-score, and recall values of 1.00, 0.83, and 0.91, respectively, indicating its proficiency in accurately recognizing healthy samples.

In contrast, the Random Forest Classifier (RFC) exhibited overall lower performance compared to SVM, with recall values for most classes being notably lower. Notably, RFC performed comparatively well in recognizing healthy samples but struggled in correctly identifying the other classes, particularly in the Premalignant category.

While the Support Vector Machine (SVM) and Random Forest Classifier (RFC) demonstrated strong performance in classifying pancreatic cancer cases, the Deep Neural Network (DNN) exhibited a comparatively lower accuracy. This discrepancy in accuracy led to the decision not to calculate class-specific recall, F1 score, and precision values for DNN. Instead, we presented average values for these metrics across the DNN model. This suggests that while DNN may have potential, it requires further optimization to match the classification accuracy achieved by SVM and RFC in the context of our study.

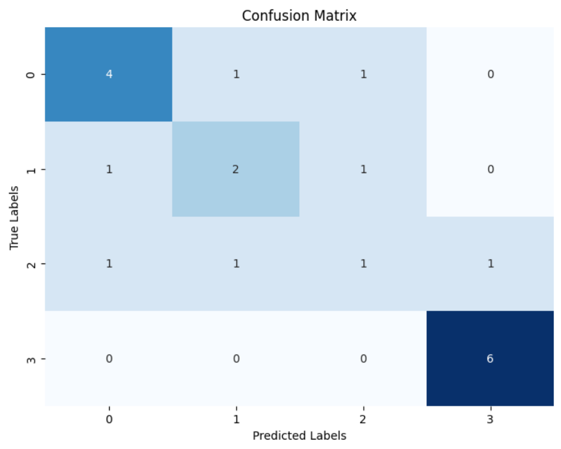


Fig 5.1-Confusion Matrix for RFC

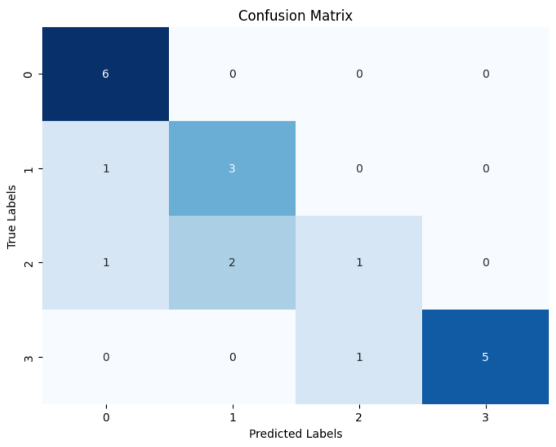


Fig 5.2-Confusion Matrix for RFC

0-PDAC 1-Benign 2-Premalignant 3-Healthy

The confusion matrix for the SVM model revealed accurate predictions for the PDAC class, where the true and predicted numbers matched perfectly. However, in the case of healthy samples, out of 6 instances, 5 were correctly predicted. On the other hand, the RFC model correctly predicted all healthy samples, but only 5 out of 6 PDAC cases were accurately classified.

When we assess the prediction performance for the Benign and Premalignant classes, it appears to be less satisfactory compared to PDAC and Healthy for both models. Nevertheless, it's worth noting that SVM demonstrated superior prediction accuracy for the Benign class compared to RFC. This suggests that SVM exhibits a higher proficiency in distinguishing Benign cases. Nevertheless, the absence of a confusion matrix for the Deep Neural Network (DNN) model is attributed to its notably lower accuracy of 0.55. Given its comparatively weaker performance in relation to the other models, it was deemed unnecessary to compare its confusion matrix with the higher-performing models, as it exhibited a clear difference in performance.

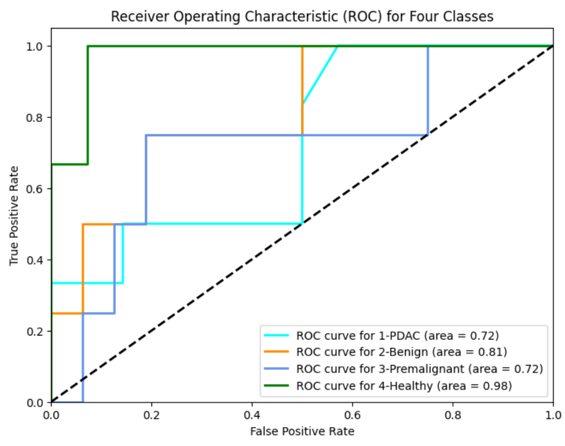


Fig5.3-ROC curve for RFC.

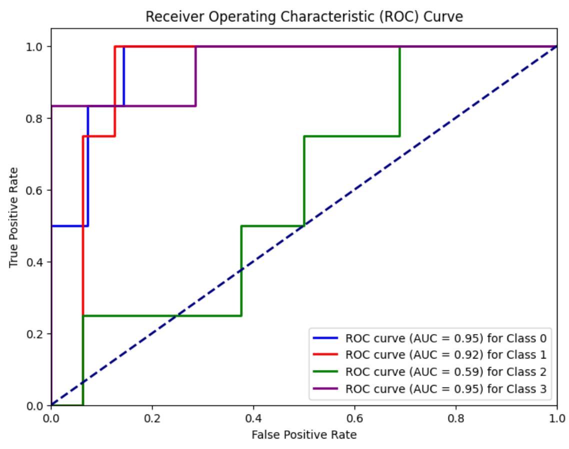


Fig5.4-ROC curve for SVC.

0-PDAC 1-Benign 2-Premalignant 3-Healthy

Drawing insights from the analysis of ROC curves depicted in Figures 5.3 and 5.4, a clear picture emerges regarding the area under the ROC curve (AUC) across different classes. The SVM model demonstrates its prowess with higher AUC values in both the PDAC and Benign classes, showcasing its effectiveness in discriminating between these categories compared to RFC. Intriguingly, the Premalignant class presents a reversal in performance, where RFC exhibits superior AUC, indicating its proficiency in handling this particular challenging category. Meanwhile, in the Healthy class, both models offer comparable performance, though RFC maintains a slight edge. This distinction is further underscored by the substantial AUC gap in the Premalignant class, with RFC achieving an impressive 0.72, while SVM trails with an AUC of 0.59, emphasizing the nuances in performance across different disease categories.

5.1 Challenges

Facing a task with a dataset comprising 1172 features and only 100 instances present several notable challenges, especially when training DNN models. The main concern here lies in the high dimensionality of the data compared to the limited number of samples. This situation can lead to issues related to overfitting, where the model may struggle to generalize well to unseen data. With such a small number of instances relative to features, the risk of the model learning noise or specific patterns in the training data, rather than true underlying relationships, is significantly elevated**.**

I employed two techniques, feature selection and cross-validation, to address these challenges. However, it's worth noting that these processes can be quite time-consuming and demanding in terms of computational resources. For instance, feature selection required parallelization to obtain results, and even with parallel processing, it took approximately 3 hours to complete. Similarly, when implementing bootstrapping, the system's RAM approached its maximum capacity due to the computational demands of the task.
Additionally, it is noteworthy that essential attributes, specifically age group, ethnicity, and gender, which have been highlighted as crucial factors in the early detection of pancreatic cancer in studies conducted by (Chen and Jeong, 2007), are not included in this dataset. The absence of these attributes poses a challenge, potentially limiting the dataset's ability to comprehensively capture the various factors that influence early detection.

6. Conclusions and further work

In this research, our aim was to comprehensively assess three distinct machine learning algorithms: Deep Neural Network (DNN), Support Vector Machine (SVM), and Random Forest Classifier (RFC) for the task of detecting pancreatic cancer. We leveraged PESI-MS data from both positive and negative modes of plasma samples to gain valuable insights into their individual performances. Among these algorithms, SVM emerged as the standout performer, achieving an accuracy of 0.75 and excelling in classifying samples across diverse categories, with notable success in identifying pancreatic cancer (PDAC) and healthy cases. RFC, while achieving an accuracy of 0.65, encountered challenges in correctly identifying most classes, except for healthy samples. On the other hand, DNN lagged with an accuracy of 0.55, signaling the need for further optimization.

Taking a closer look at SVM's performance at the class level, we observed strengths in accurately identifying PDAC and healthy cases, although it faced challenges in the Premalignant class. RFC exhibited lower recall values overall, pointing out its limitations in certain categories. The analysis of confusion matrices and ROC curves provided deeper insights into the models' performance, highlighting nuanced differences across disease categories. SVM demonstrated its prowess with higher AUC values in PDAC and Benign classes, while RFC excelled in the Premalignant category.

Regarding further work, there are several promising avenues to explore. These include refining the DNN model to enhance its classification performance, employing data augmentation techniques to address limited dataset size and high dimensionality, incorporating crucial attributes like age group, ethnicity, and gender to capture diverse factors influencing early detection, investigating ensemble approaches for improved classification results, validating the models through clinical trials and real-world datasets, and optimizing computational resources to streamline research efforts. In summary, while this research has illuminated the potential of machine learning in pancreatic cancer detection, it underscores the importance of ongoing efforts to enhance model performance and real world applicability.

Moving forward, future research in this domain should consider exploring specific biomarkers for early stage pancreatic cancer detection. Identifying biomarkers tailored to early detection can significantly enhance diagnostic accuracy and intervention effectiveness. This endeavor involves a thorough investigation into potential biomarkers and the application of advanced machine learning techniques for their identification. Additionally, validation of these biomarkers through clinical trials and their translation into clinical practice will be crucial in improving early detection and ultimately enhancing patient outcomes in the battle against pancreatic cancer.

7.Self evaluation

At the outset of this research project, I initially aimed to achieve an accuracy of 80% using the Deep Neural Network (DNN) algorithm. However, as I delved into the dataset, I encountered challenges that significantly impacted my expectations. The dataset proved to be unbalanced and relatively small in size, which led me to realize that my target accuracy was more challenging to attain than initially anticipated. This realization prompted me to reevaluate my approach.

Recognizing the complexity of the task at hand, I proactively sought guidance from my professor, engaging in weekly discussions to gain insights and refine my research strategy. These interactions were instrumental in my academic growth, as I learned how to develop a robust machine learning model by starting with a simple foundation and progressively building upon it. My professor's mentorship provided me with a clear understanding of the entire process involved in constructing a machine learning model and writing an academically sound dissertation.

Through this journey, I developed not only my technical skills but also a deeper appreciation for the intricacies of data analysis and machine learning. I realized the importance of adapting to the unique characteristics of the dataset and the significance of seeking guidance from experienced mentors. This experience has not only enhanced my academic abilities but also equipped me with valuable problem-solving skills and the ability to navigate complex research tasks effectively. It underscores the importance of continuous learning and collaboration in the pursuit of academic excellence.

8.References

Chung WY, Correa E, Yoshimura K, Chang MC, Dennison A, Takeda S, Chang YT. Using probe electrospray ionization mass spectrometry and machine learning for detecting pancreatic cancer with high performance. Am J Transl Res. 2020 Jan 15;12(1):171-179. PMID: 32051746; PMCID: PMC7013221.

Iwano T, Yoshimura K, Watanabe G, Saito R, Kiritani S, Kawaida H, Moriguchi T, Murata T, Ogata K, Ichikawa D, Arita J, Hasegawa K, Takeda S. High-performance Collective Biomarker from Liquid Biopsy for Diagnosis of Pancreatic Cancer Based on Mass Spectrometry and Machine Learning. J Cancer. 2021 Nov 4;12(24):7477-7487. doi: 10.7150/jca.63244. PMID: 35003367; PMCID: PMC8734412.

Preuss, K.; Thach, N.; Liang, X.; Baine, M.; Chen, J.; Zhang, C.; Du, H.; Yu, H.; Lin, C.; Hollingsworth, M.A.; Zheng, D. Using Quantitative Imaging for Personalized Medicine in Pancreatic Cancer: A Review of Radiomics and Deep Learning Applications. Cancers 2022, 14, 1654. https://doi.org/10.3390/cancers14071654

Bakasa, W. and Viriri, S. (2021) 'Pancreatic Cancer Survival Prediction: A Survey of the State-of-the-Art', *Computational and Mathematical Methods in Medicine*. Edited by H. Chen, 2021, pp. 1–17. Available at: https://doi.org/10.1155/2021/1188414.

Bordag, N. *et al.* (no date) 'Towards fast, routine blood sample quality evaluation by Probe Electrospray Ionization (PESI) metabolomics'.

Chen, X. and Jeong, J.C. (2007) 'Enhanced recursive feature elimination', in *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, Cincinnati, OH, USA: IEEE, pp. 429–435. Available at: https://doi.org/10.1109/ICMLA.2007.35.

Conroy, M.C. *et al.* (2023) 'UK Biobank: a globally important resource for cancer research', *British Journal of Cancer*, 128(4), pp. 519–527. Available at: https://doi.org/10.1038/s41416-022-02053-5.

Nagarajan, G. and Dhinesh Babu, L.D. (2022) 'Missing data imputation on biomedical data using deeply learned clustering and L2 regularized regression based on symmetric uncertainty', *Artificial Intelligence in Medicine*, 123, p. 102214. Available at: https://doi.org/10.1016/j.artmed.2021.102214.

Potdar, K., S., T. and D., C. (2017) 'A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers', *International Journal of Computer Applications*, 175(4), pp. 7–9. Available at: https://doi.org/10.5120/ijca2017915495.

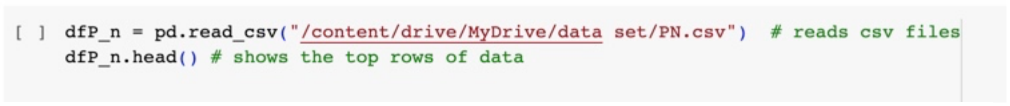
Takeda, S., Yoshimura, K. and Tanihata, H. (2020) 'Sample Preparation for Probe Electrospray Ionization Mass Spectrometry', *Journal of Visualized Experiments*, (156), p. 59942. Available at: https://doi.org/10.3791/59942.

Yacouby, R. and Axman, D. (2020) ‘Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models’, in *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*. *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, Online: Association for Computational Linguistics, pp. 79–91. Available at: https://doi.org/10.18653/v1/2020.eval4nlp-1.9.

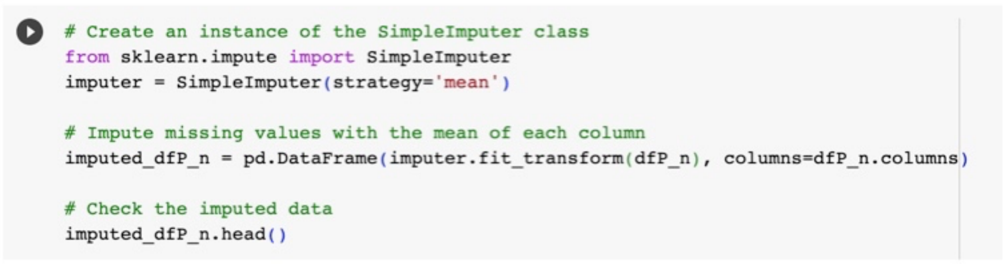
Tovar DR, Rosenthal MH, Maitra A, Koay EJ. Potential of artificial intelligence in the risk stratification for and early detection of pancreatic cancer. Art Int Surg 2023;3:14-26. https://dx.doi.org/10.20517/ais.2022.38

9.Appendix

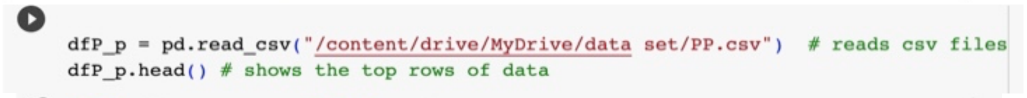
9.1 Data preprocessing



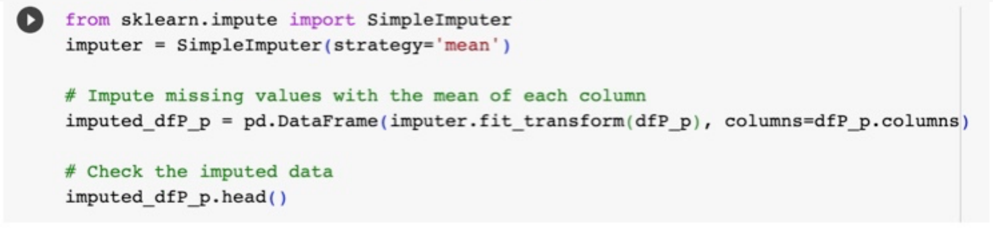
```
dfP_n = pd.read_csv("/content/drive/MyDrive/data set/PN.csv")  # reads csv files
dfP_n.head() # shows the top rows of data
```



```
# Create an instance of the SimpleImputer class
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')

# Impute missing values with the mean of each column
imputed_dfP_n = pd.DataFrame(imputer.fit_transform(dfP_n), columns=dfP_n.columns)

# Check the imputed data
imputed_dfP_n.head()
```

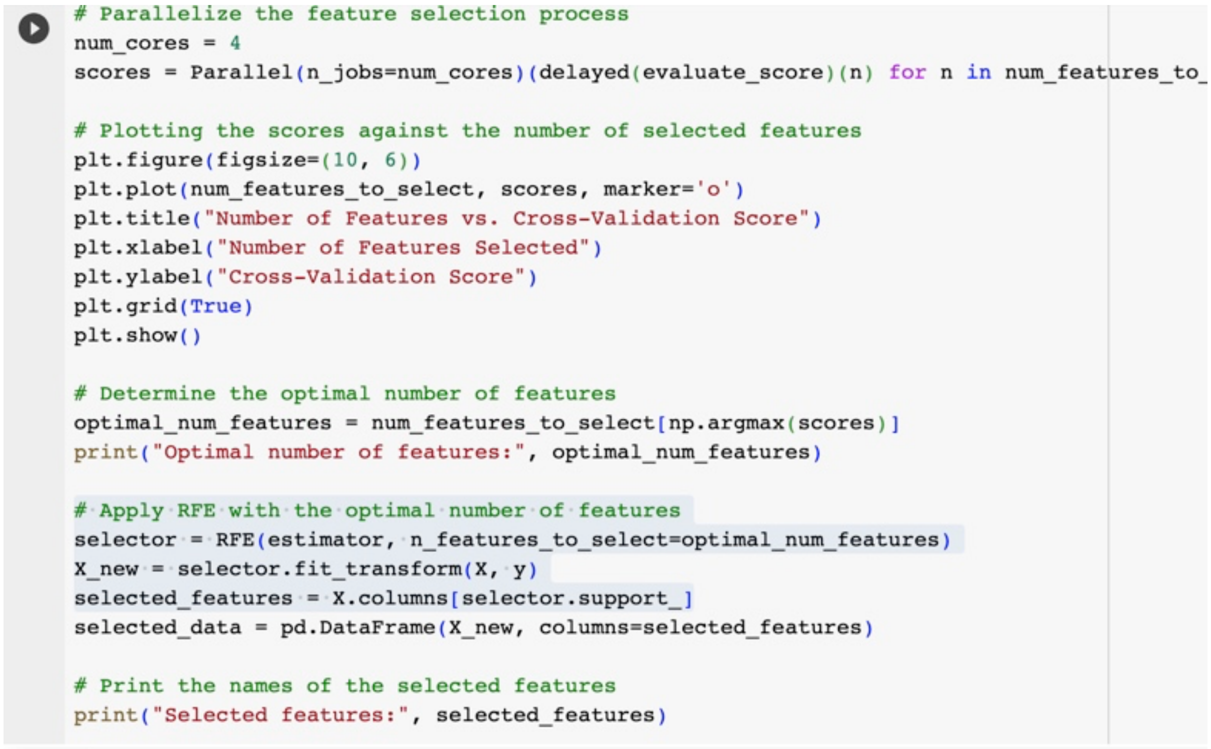


```
dfP_p = pd.read_csv("/content/drive/MyDrive/data set/PP.csv")  # reads csv files
dfP_p.head() # shows the top rows of data
```

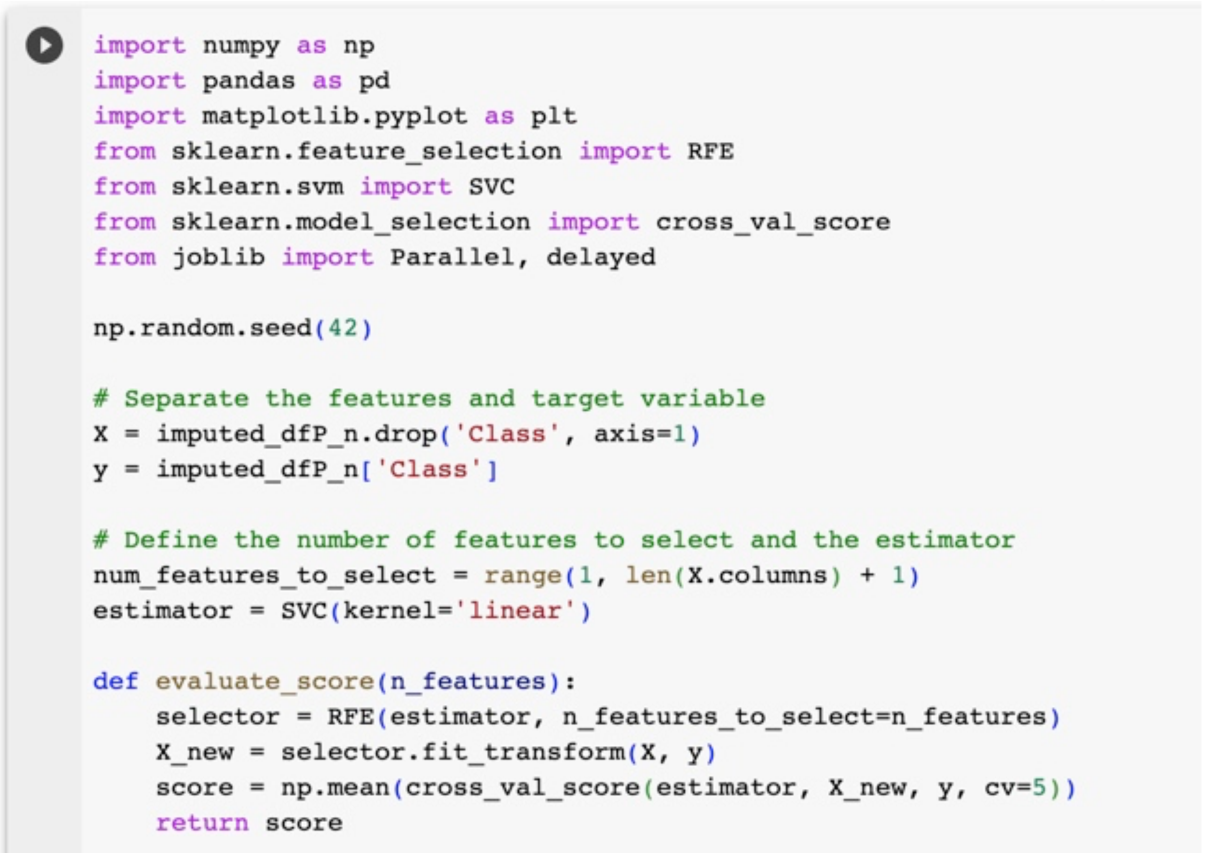


```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')

# Impute missing values with the mean of each column
imputed_dfP_p = pd.DataFrame(imputer.fit_transform(dfP_p), columns=dfP_p.columns)

# Check the imputed data
imputed_dfP_p.head()
```

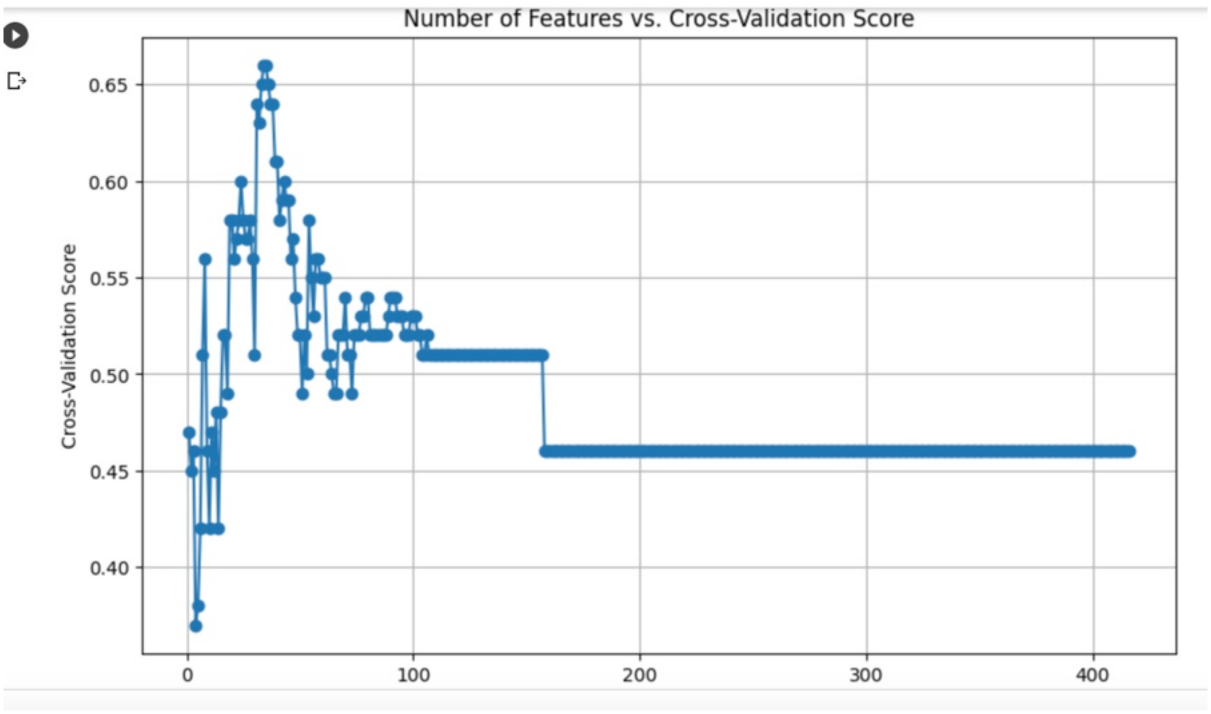
9.1 Feature selection



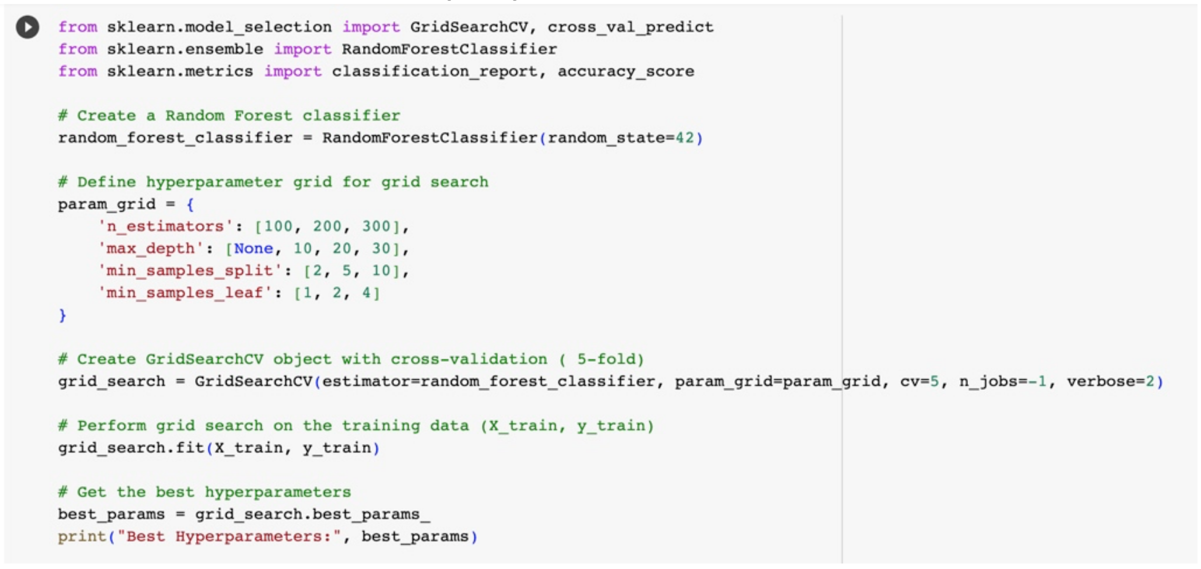
```
# Parallelize the feature selection process
num_cores = 4
scores = Parallel(n_jobs=num_cores)(delayed(evaluate_score)(n) for n in num_features_to_

# Plotting the scores against the number of selected features
plt.figure(figsize=(10, 6))
plt.plot(num_features_to_select, scores, marker='o')
plt.title("Number of Features vs. Cross-Validation Score")
plt.xlabel("Number of Features Selected")
plt.ylabel("Cross-Validation Score")
plt.grid(True)
plt.show()

# Determine the optimal number of features
optimal_num_features = num_features_to_select[np.argmax(scores)]
print("Optimal number of features:", optimal_num_features)

# Apply RFE with the optimal number of features
selector = RFE(estimator, n_features_to_select=optimal_num_features)
X_new = selector.fit_transform(X, y)
selected_features = X.columns[selector.support_]
selected_data = pd.DataFrame(X_new, columns=selected_features)

# Print the names of the selected features
print("Selected features:", selected_features)
```



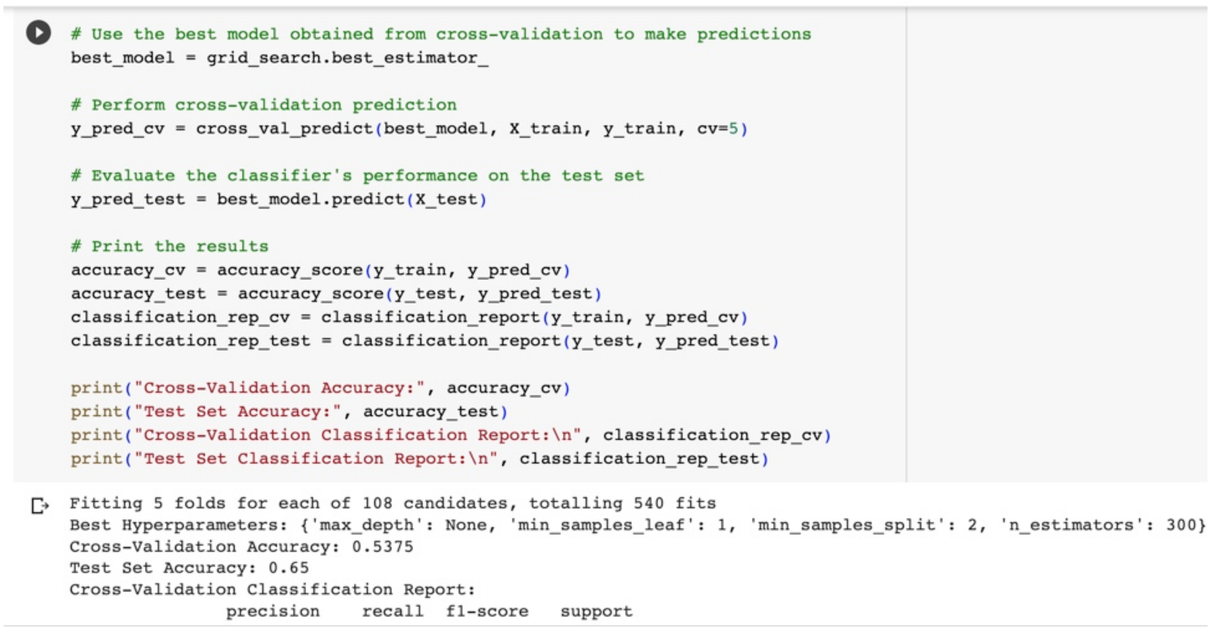
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_selection import RFE
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from joblib import Parallel, delayed

np.random.seed(42)

# Separate the features and target variable
X = imputed_dfP_n.drop('Class', axis=1)
y = imputed_dfP_n['Class']

# Define the number of features to select and the estimator
num_features_to_select = range(1, len(X.columns) + 1)
estimator = SVC(kernel='linear')

def evaluate_score(n_features):
    selector = RFE(estimator, n_features_to_select=n_features)
    X_new = selector.fit_transform(X, y)
    score = np.mean(cross_val_score(estimator, X_new, y, cv=5))
    return score
```



9.2 Random Forest Classifier (RFC)

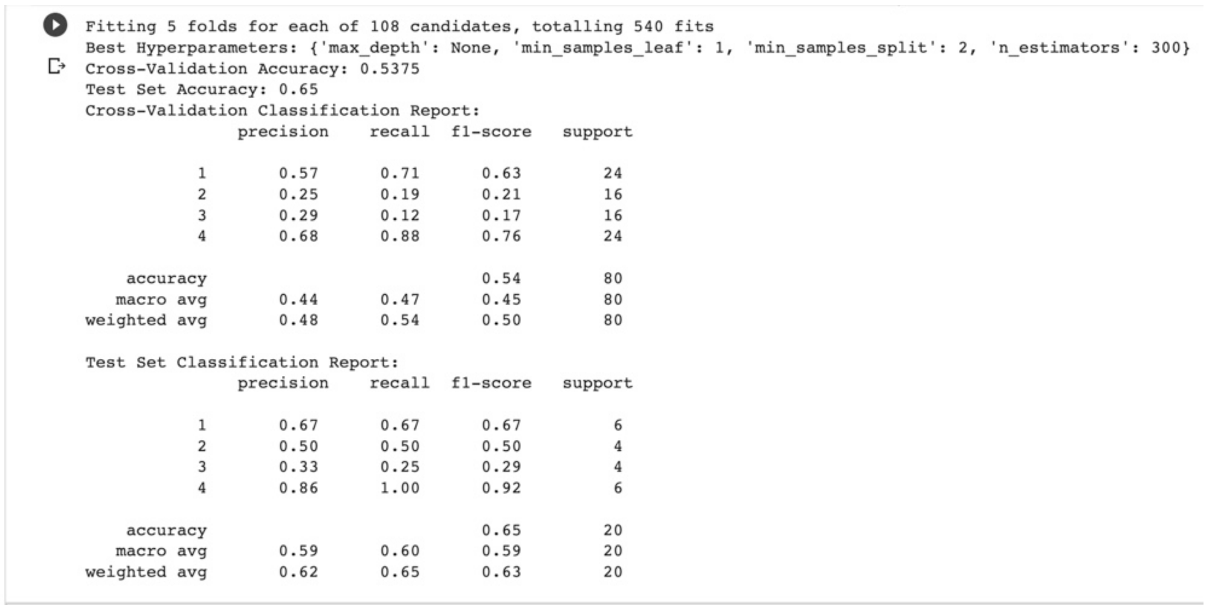


```
from sklearn.model_selection import GridSearchCV, cross_val_predict
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Create a Random Forest classifier
random_forest_classifier = RandomForestClassifier(random_state=42)

# Define hyperparameter grid for grid search
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create GridSearchCV object with cross-validation ( 5-fold)
grid_search = GridSearchCV(estimator=random_forest_classifier, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)

# Perform grid search on the training data (X_train, y_train)
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)
```



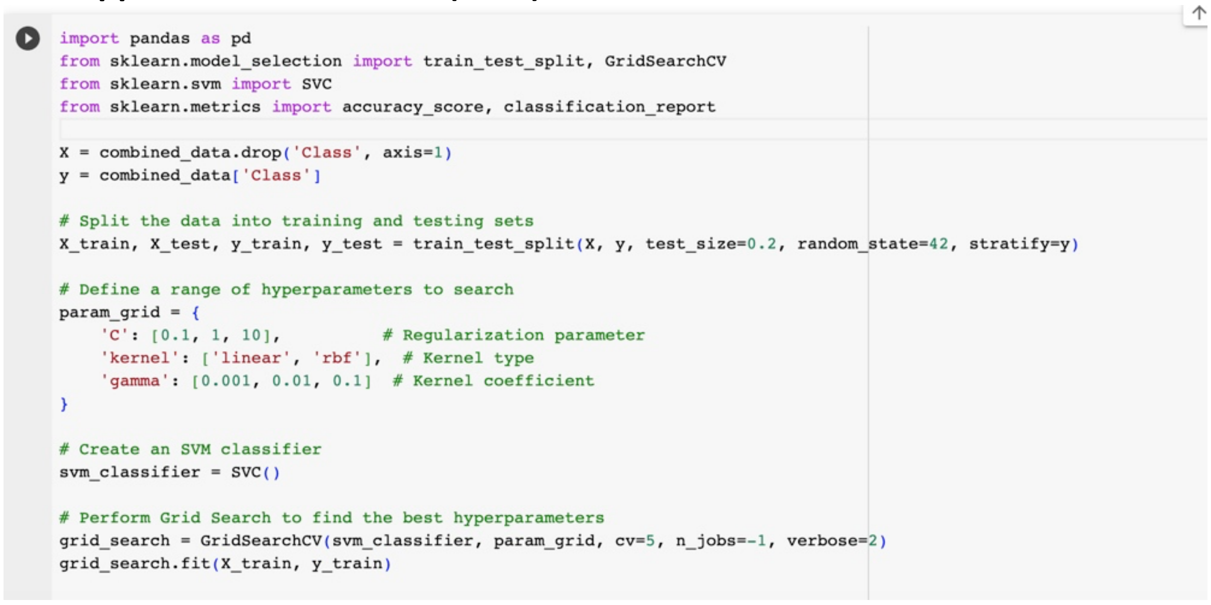
```python
# Use the best model obtained from cross-validation to make predictions
best_model = grid_search.best_estimator_

# Perform cross-validation prediction
y_pred_cv = cross_val_predict(best_model, X_train, y_train, cv=5)

# Evaluate the classifier's performance on the test set
y_pred_test = best_model.predict(X_test)

# Print the results
accuracy_cv = accuracy_score(y_train, y_pred_cv)
accuracy_test = accuracy_score(y_test, y_pred_test)
classification_rep_cv = classification_report(y_train, y_pred_cv)
classification_rep_test = classification_report(y_test, y_pred_test)

print("Cross-Validation Accuracy:", accuracy_cv)
print("Test Set Accuracy:", accuracy_test)
print("Cross-Validation Classification Report:\n", classification_rep_cv)
print("Test Set Classification Report:\n", classification_rep_test)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
Cross-Validation Accuracy: 0.5375
Test Set Accuracy: 0.65
Cross-Validation Classification Report:
               precision    recall  f1-score   support
```

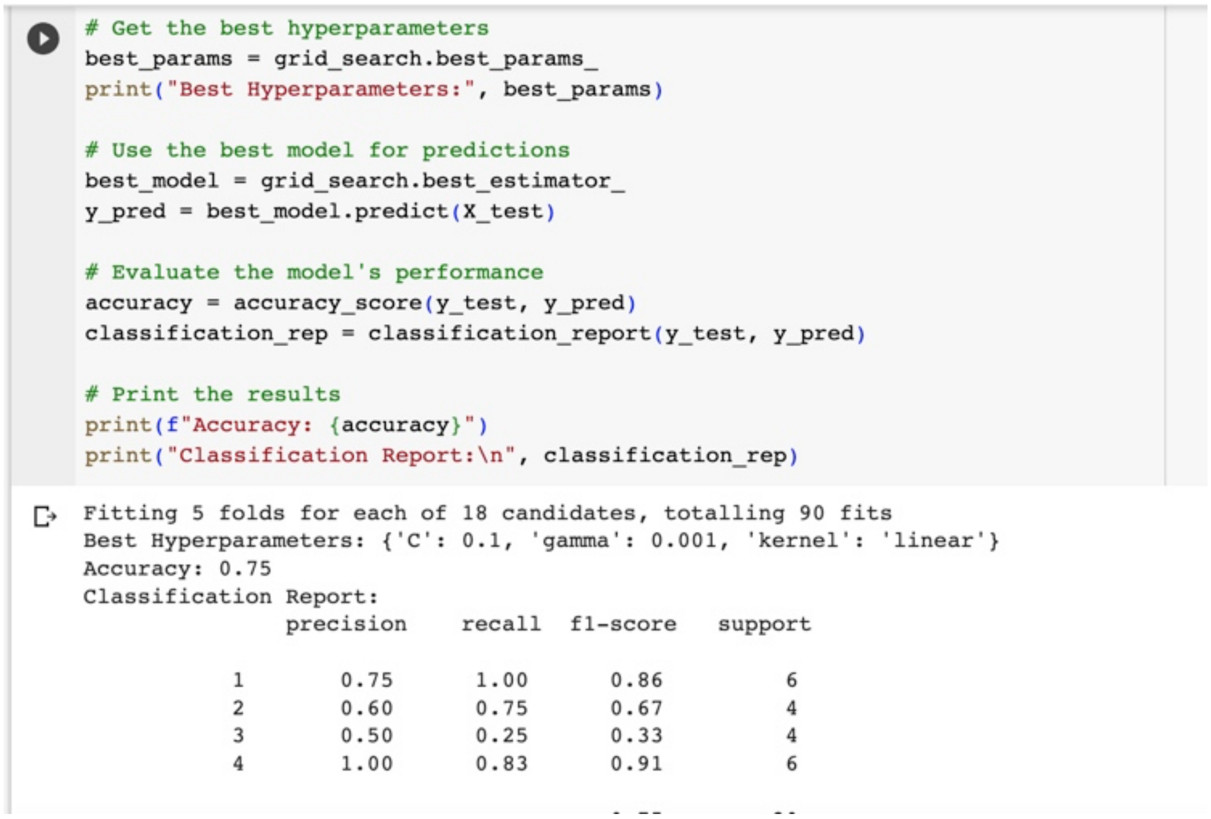


```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
Cross-Validation Accuracy: 0.5375
Test Set Accuracy: 0.65
Cross-Validation Classification Report:
               precision    recall  f1-score   support

           1       0.57      0.71      0.63        24
           2       0.25      0.19      0.21        16
           3       0.29      0.12      0.17        16
           4       0.68      0.88      0.76        24

    accuracy                           0.54        80
   macro avg       0.44      0.47      0.45        80
weighted avg       0.48      0.54      0.50        80

Test Set Classification Report:
               precision    recall  f1-score   support

           1       0.67      0.67      0.67         6
           2       0.50      0.50      0.50         4
           3       0.33      0.25      0.29         4
           4       0.86      1.00      0.92         6

    accuracy                           0.65        20
   macro avg       0.59      0.60      0.59        20
weighted avg       0.62      0.65      0.63        20
```

9.3 Support Vector Machine (SVM)



```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

X = combined_data.drop('Class', axis=1)
y = combined_data['Class']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Define a range of hyperparameters to search
param_grid = {
    'C': [0.1, 1, 10],          # Regularization parameter
    'kernel': ['linear', 'rbf'],  # Kernel type
    'gamma': [0.001, 0.01, 0.1]  # Kernel coefficient
}

# Create an SVM classifier
svm_classifier = SVC()

# Perform Grid Search to find the best hyperparameters
grid_search = GridSearchCV(svm_classifier, param_grid, cv=5, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)
```



```python
# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Use the best model for predictions
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Classification Report:\n", classification_rep)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Best Hyperparameters: {'C': 0.1, 'gamma': 0.001, 'kernel': 'linear'}
Accuracy: 0.75
Classification Report:
               precision    recall  f1-score   support

           1       0.75      1.00      0.86         6
           2       0.60      0.75      0.67         4
           3       0.50      0.25      0.33         4
           4       1.00      0.83      0.91         6
```