Est. 1841 | YORK ST JOHN UNIVERSITY

Module Title	**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**
Module code	**LDS7003M**
Student Name	**DEVANGI MIRAL KUMAR PATEL**
Student ID	**230264357**
Student Email	**Devangi.patel1@yorksj.ac.uk**
Submission Date	**18 January 2024**

Student Declaration:

I confirm that I have read and understood the University Policy and Procedures on Academic Misconduct and that the work submitted is my own.	✓
I confirm that I have processed and produced this submission in accordance with the University guidelines and the assessment brief regarding the use of generative AI. Where appropriate, I have acknowledged where and how it has been used. OR Where generative AI is not permitted in this assignment. I confirm that it has not been used in any part of the process or production of this submission https://www.yorksj.ac.uk/policies-and-documents/generative-artificial-intelligence/	✓ ✓

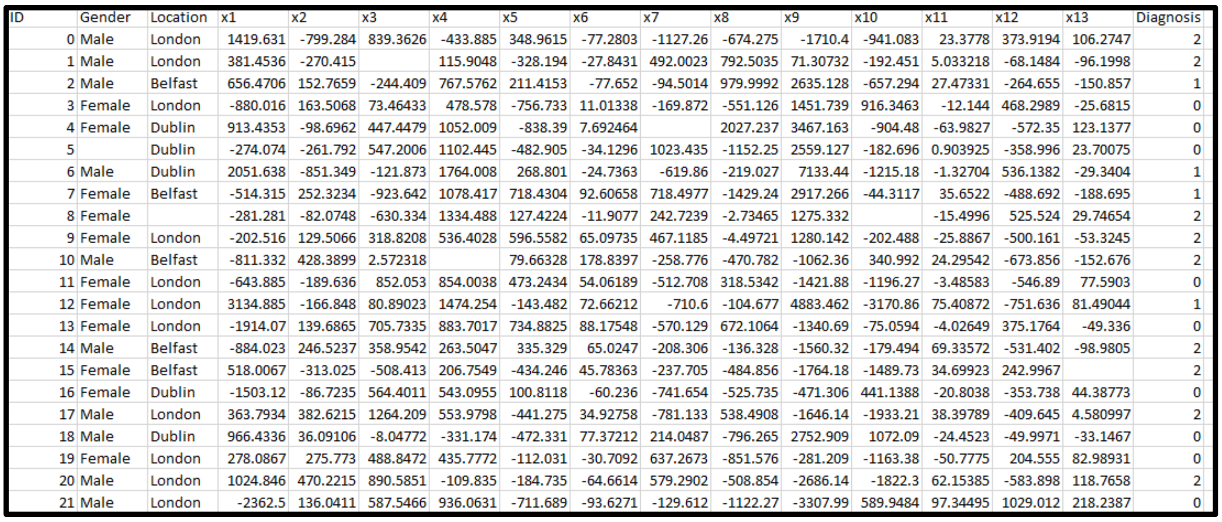
Table of Contents

1. Introduction

In this extensive investigation, two different machine-learning methods have been employed to solve the classification problem. Human diagnostic studies employed material that contained both quantities and lots. The training set was constructed operating cross-validation, which allowed to address data gaps for both quantitative and categorical variables. The three methods including feature scaling, k-fold cross-validation and categorical variable generating are seamlessly integrated. Evaluation metrics pay attention to the baseline performance of algorithms. The next stage of the study is feature selection, which involves finding the ideal number of elements and performing recursive feature elimination. Finally, the model parameters are refined by combining the proposed feature with the hyperparameter tuning technique with the help of GridSearch.

2. Data Description

The dataset utilised throughout this study consisted of several numeric variables representing different characteristics and columns containing attributes such as ID, gender, and location. Data from analysis sections is used to create certain sections. With 374 rows and 17 columns, this dataset is rather extensive.



ID	Gender	Location	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	Diagnosis
0	Male	London	1419.631	-799.284	839.3626	-433.885	348.9615	-77.2803	-1127.26	-674.275	-1710.4	-941.083	23.3778	373.9194	106.2747	2
1	Male	London	381.4536	-270.415		115.9048	-328.194	-27.8431	492.0023	792.5035	71.30732	-192.451	5.033218	-68.1484	-96.1998	2
2	Male	Belfast	656.4706	152.7659	-244.409	767.5762	211.4153	-77.652	-94.5014	979.9992	2635.128	-657.294	27.47331	-264.655	-150.857	1
3	Female	London	-880.016	163.5068	73.46433	478.578	-756.733	11.01338	-169.872	-551.126	1451.739	916.3463	-12.144	468.2989	-25.6815	0
4	Female	Dublin	913.4353	-98.6962	447.4479	1052.009	-838.39	7.692464		2027.237	3467.163	-904.48	-63.9827	-572.35	123.1377	0
5		Dublin	-274.074	-261.792	547.2006	1102.445	-482.905	-34.1296	1023.435	-1152.25	2559.127	-182.696	0.903925	-358.996	23.70075	0
6	Male	Dublin	2051.638	-851.349	-121.873	1764.008	268.801	-24.7363	-619.86	-219.027	7133.44	-1215.18	-1.32704	536.1382	-29.3404	1
7	Female	Belfast	-514.315	252.3234	-923.642	1078.417	718.4304	92.60658	718.4977	-1429.24	2917.266	-44.3117	35.6522	-488.692	-188.695	1
8	Female		-281.281	-82.0748	-630.334	1334.488	127.4224	-11.9077	242.7239	-2.73465	1275.332		-15.4996	525.524	29.74654	2
9	Female	London	-202.516	129.5066	318.8208	536.4028	596.5582	65.09735	467.1185	-4.49721	1280.142	-202.488	-25.8867	-500.161	-53.3245	2
10	Male	Belfast	-811.332	428.3899	2.572318		79.66328	178.8397	-258.776	-470.782	-1062.36	340.992	24.29542	-673.856	-152.676	2
11	Female	London	-643.885	-189.636	852.053	854.0038	473.2434	54.06189	-512.708	318.5342	-1421.88	-1196.27	-3.48583	-546.89	77.5903	0
12	Female	London	3134.885	-166.848	80.89023	1474.254	-143.482	72.66212	-710.6	-104.677	4883.462	-3170.86	75.40872	-751.636	81.49044	1
13	Female	London	-1914.07	139.6865	705.7335	883.7017	734.8825	88.17548	-570.129	672.1064	-1340.69	-75.0594	-4.02649	375.1764	-49.336	0
14	Male	Belfast	-884.023	246.5237	358.9542	263.5047	335.329	65.0247	-208.306	-136.328	-1560.32	-179.494	69.33572	-531.402	-98.9805	2
15	Female	Belfast	518.0067	-313.025	-508.413	206.7549	-434.246	45.78363	-237.705	-484.856	-1764.18	-1489.73	34.69923	242.9967		2
16	Female	Dublin	-1503.12	-86.7235	564.4011	543.0955	100.8118	-60.236	-741.654	-525.735	-471.306	441.1388	-20.8038	-353.738	44.38773	0
17	Male	London	363.7934	382.6215	1264.209	553.9798	-441.275	34.92758	-781.133	538.4908	-1646.14	-1933.21	38.39789	-409.645	4.580997	2
18	Male	Dublin	966.4336	36.09106	-8.04772	-331.174	-472.331	77.37212	214.0487	-796.265	2752.909	1072.09	-24.4523	-49.9971	-33.1467	0
19	Female	London	278.0867	275.773	488.8472	435.7772	-112.031	-30.7092	637.2673	-851.576	-281.209	-1163.38	-50.7775	204.555	82.98931	0
20	Male	London	1024.846	470.2215	890.5851	-109.835	-184.735	-64.6614	579.2902	-508.854	-2686.14	-1822.3	62.15385	-583.898	118.7658	2
21	Male	London	-2362.5	136.0411	587.5466	936.0631	-711.689	-93.6271	-129.612	-1122.27	-3307.99	589.9484	97.34495	1029.012	218.2387	0

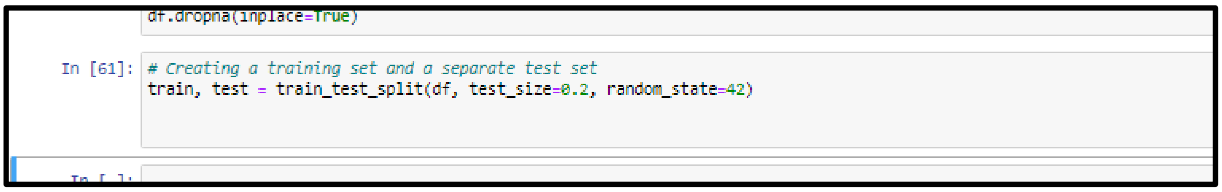
Figure 1: Components of the dataset

Fifteen of these columns contain numerical data, while the other two house object values. Unlike the categorical data provided by gender and location, numeric variables have many possible values. The dataset contains very little information and should be used with caution. The collection is geographically diverse thanks to many people from London, Belfast, and Dublin. The goal of data mining research is to make analytical predictions based on given characteristics. Analysing and organizing data is what the machine learning process of data analysis is all about.

3. Data Pre-processing

3.1 Creating Training and Testing Sets

Data preparation is an important step in making machine learning models efficient and scalable (Cioffi *et al*. 2020). It all starts with the selection of independent and target variables. The diagnosis column has been chosen as the target variable to forecast the status of the diagnosis of the people based on several independent factors. In order to simplify model training and evaluation and reduce redundancy, after the dataset is compiled, subsets of the dataset are created for testing and training.



```
df.dropna(inplace=True)

In [61]: # Creating a training set and a separate test set
         train, test = train_test_split(df, test_size=0.2, random_state=42)
```

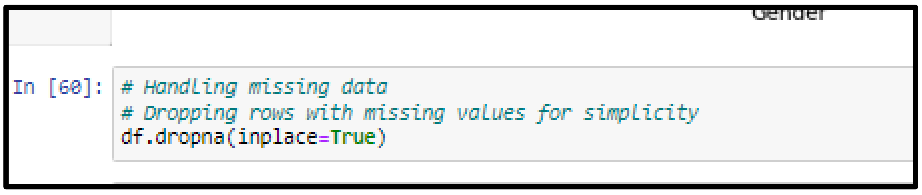
Figure 2: Generation of training and testing sets

(Source: Developed from Jupyter Notebook)

The above figure depicts the creation of train and test sets of the data to fit the ML models for prediction.

3.2 Handling Missing Data

Dealing with empty sets Data integrity is maintained by filling in gaps with missing or incorrect data via interpolation or deletion.



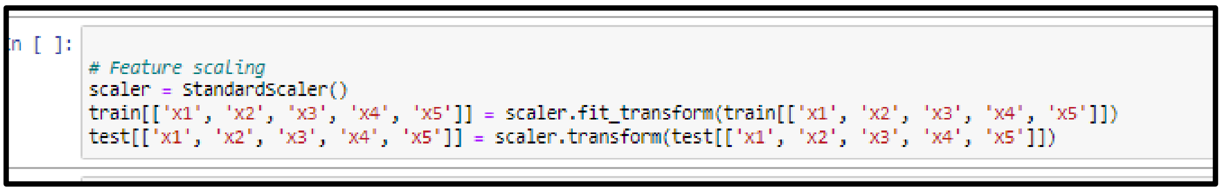
```
In [60]: # Handling missing data
         # Dropping rows with missing values for simplicity
         df.dropna(inplace=True)
```

Figure 3: Removing null values for both numeric and object columns

(Source: Developed from Jupyter Notebook)

It is found that the ‘dropna()’ function has been applied to eliminate null arguments from the selected dataset to maintain the quality of the analysis.

3.3 Feature Scaling



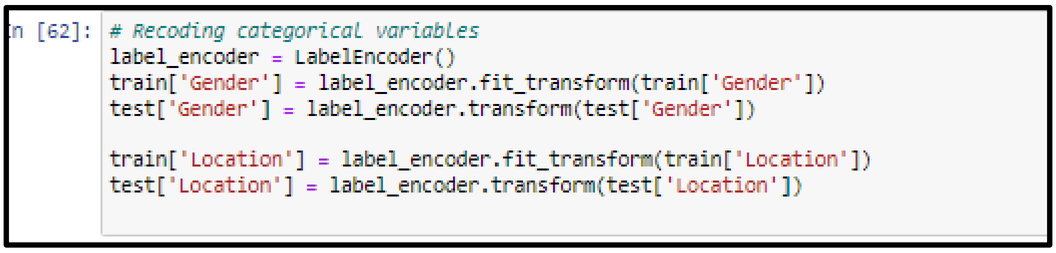
```
In [ ]:
        # Feature scaling
        scaler = StandardScaler()
        train[['x1', 'x2', 'x3', 'x4', 'x5']] = scaler.fit_transform(train[['x1', 'x2', 'x3', 'x4', 'x5']])
        test[['x1', 'x2', 'x3', 'x4', 'x5']] = scaler.transform(test[['x1', 'x2', 'x3', 'x4', 'x5']])
```

Figure 4: Feature scaling based on the training and test set

(Source: Developed from Jupyter Notebook)

Feature scaling is fundamental to guarantee predictable scales among highlights (Koçak *et al*. 2023). Utilizing StandardScaler, 'x1' to 'x5' in both the preparation and test sets are normalised. This interaction further develops calculation convergence and discourages the strength of specific highlights because of their larger scopes, fostering better model execution.

3.4 Recoding Categorical Variables



```
In [62]: # Recoding categorical variables
         label_encoder = LabelEncoder()
         train['Gender'] = label_encoder.fit_transform(train['Gender'])
         test['Gender'] = label_encoder.transform(test['Gender'])

         train['Location'] = label_encoder.fit_transform(train['Location'])
         test['Location'] = label_encoder.transform(test['Location'])
```

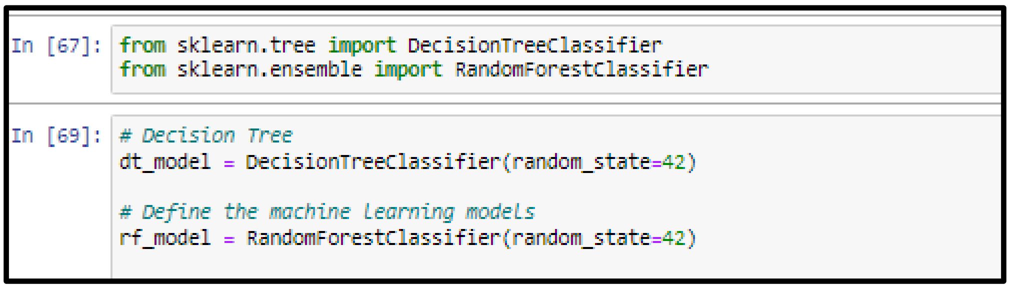
Figure 5: Recoding of categorical variables

(Source: Developed from Jupyter Notebook)

Within the "Recoding of categorical factors" step, here employ a Label Encoder to convert categorical factors into numerical representations. The 'Gender' and 'Location' columns are recoded, mapping each one-of-a-kind category to compare numbers. This handle is pivotal for machine learning calculations, which regularly require numerical input. It occasion, the 'Gender' column, initially containing categories like 'Male' and 'Female,' is presently spoken to as numerical names This change guarantees compatibility with calculations that work on numerical information, encouraging the model-building handle and enabling compelling classification based on categorical highlights within the dataset.

4. Machine Learning Algorithms

4.1 Selection of Algorithms



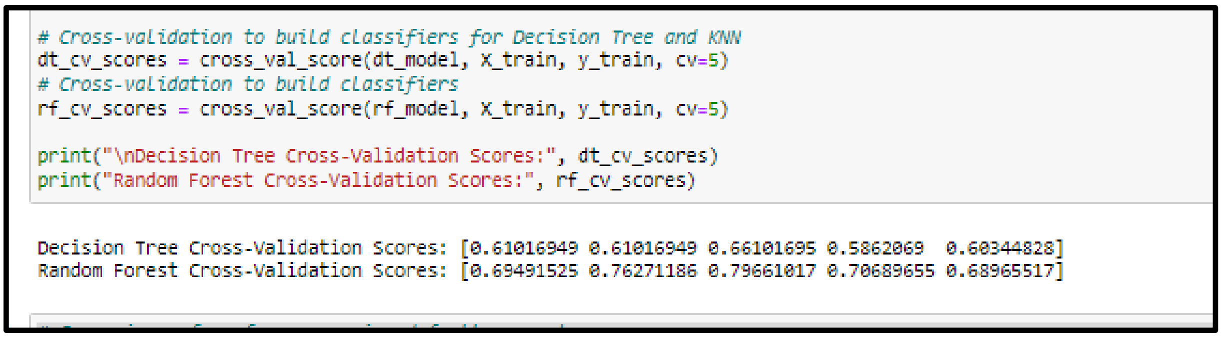
```
In [67]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import RandomForestClassifier

In [69]: # Decision Tree
         dt_model = DecisionTreeClassifier(random_state=42)

         # Define the machine learning models
         rf_model = RandomForestClassifier(random_state=42)
```

Figure 6: Choosing a decision tree and random forest classification model

(Source: Developed from Jupyter Notebook)

The "Decision Tree and Random Forest" are chosen as two grouping calculations for their particular attributes. The Decision Tree, addressed by 'dt_model', is a straightforward, interpretable model that recursively parts the dataset given highlights, making a tree-like design. Then again, the Random Forest meant as 'rf_model', is a group of different Decision Tree, presenting variety and alleviating overfitting. It consolidates the prescient force of individual trees, and Random Forest upgrades strength and speculation, making it a strong decision for complex datasets. The two models are introduced with a proper random state (random_state=42) for reproducibility in results.

4.2 Cross-validation for Model Building



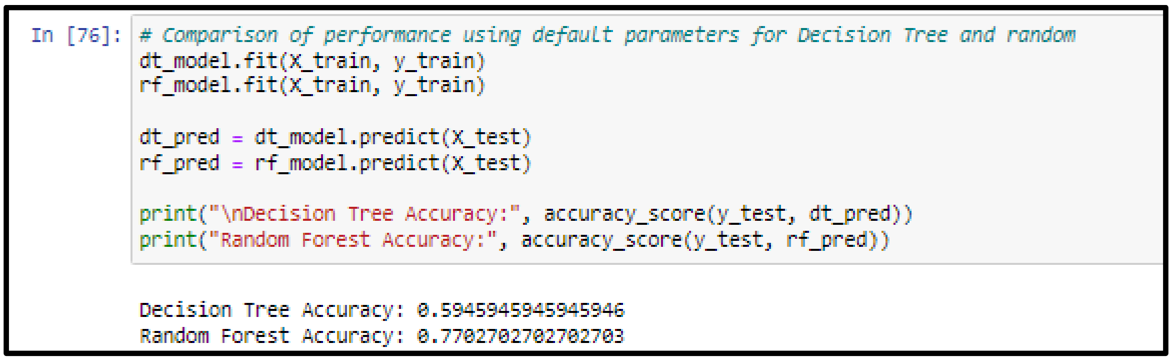
```
# Cross-validation to build classifiers for Decision Tree and KNN
dt_cv_scores = cross_val_score(dt_model, X_train, y_train, cv=5)
# Cross-validation to build classifiers
rf_cv_scores = cross_val_score(rf_model, X_train, y_train, cv=5)

print("\nDecision Tree Cross-Validation Scores:", dt_cv_scores)
print("Random Forest Cross-Validation Scores:", rf_cv_scores)

Decision Tree Cross-Validation Scores: [0.61016949 0.61016949 0.66101695 0.5862069  0.60344828]
Random Forest Cross-Validation Scores: [0.69491525 0.76271186 0.79661017 0.70689655 0.68965517]
```

Figure 7: Applying cross-validation to the ML models

(Source: Developed from Jupyter Notebook)

Within the cross-validation preparation, Decision Tree and Random Forest classifiers were assessed on the preparing set employing a 5-fold cross-validation approach. The Decision Tree showed a normal precision of roughly 61%, demonstrating direct execution over folds. On the other hand, the Random Forest outflanked the Decision Tree with a normal precision of around 71%, recommending superior generalization and vigour. These scores give bits of knowledge into the models' execution amid preparing, directing encouraging investigation, and optimization steps. The Random Forest's higher cross-validation scores indicate its potential for predominant prescient capability on concealed information compared to the Decision Tree.

4.3 Comparison of Algorithms performance using default parameters



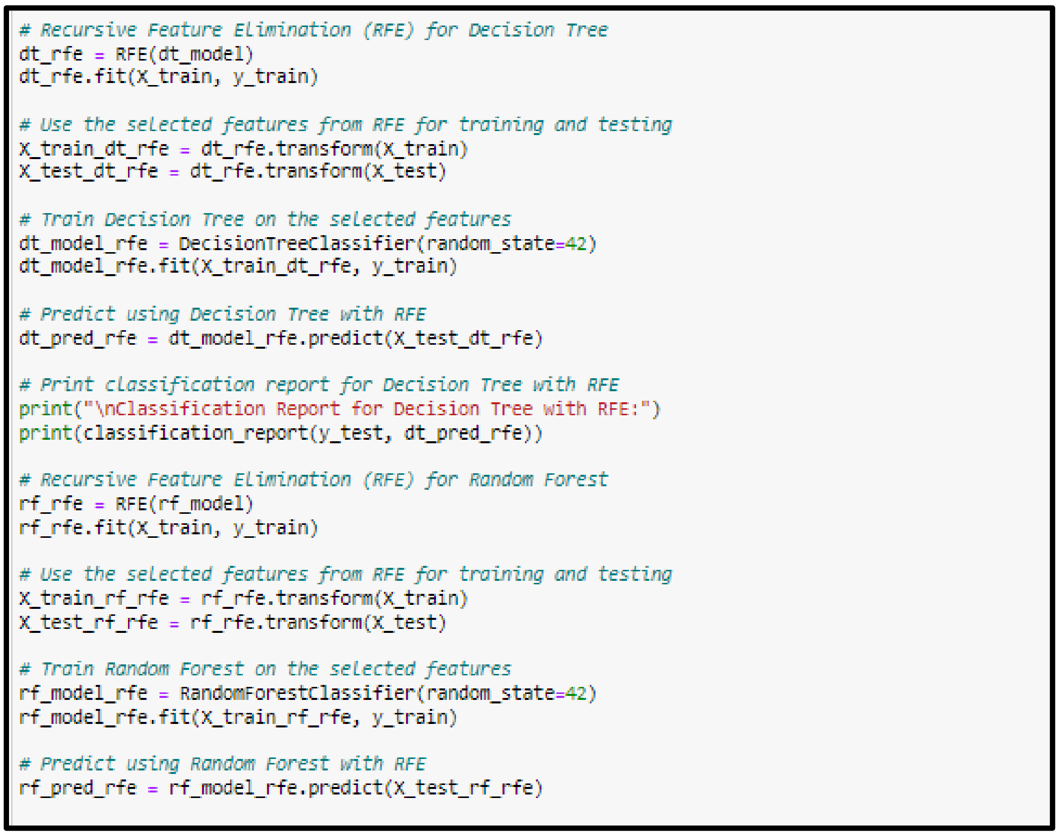
```
In [76]: # Comparison of performance using default parameters for Decision Tree and random
         dt_model.fit(X_train, y_train)
         rf_model.fit(X_train, y_train)

         dt_pred = dt_model.predict(X_test)
         rf_pred = rf_model.predict(X_test)

         print("\nDecision Tree Accuracy:", accuracy_score(y_test, dt_pred))
         print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))


         Decision Tree Accuracy: 0.5945945945945946
         Random Forest Accuracy: 0.7702702702702703
```

Figure 8: Identification of model efficacy with accuracy score

(Source: Developed from Jupyter Notebook)

Decision trees establish relationships between attributes and outcomes to explain and clarify classification (Helm *et al*. 2020). Random Forest, a collection of decision trees, improves accuracy by comparing and adjusting lower and upper bounds for padding. Simple and reliable classification methods provide an excellent ability to handle complex datasets with variable characteristics. Both models are fitted successfully in the train sets to predict the future outcomes in the test sets. It has been observed that the accuracy scores of decision trees as well as random forest models are 59% and 77% respectively.

4.4 Recursive Feature Elimination (RFE)



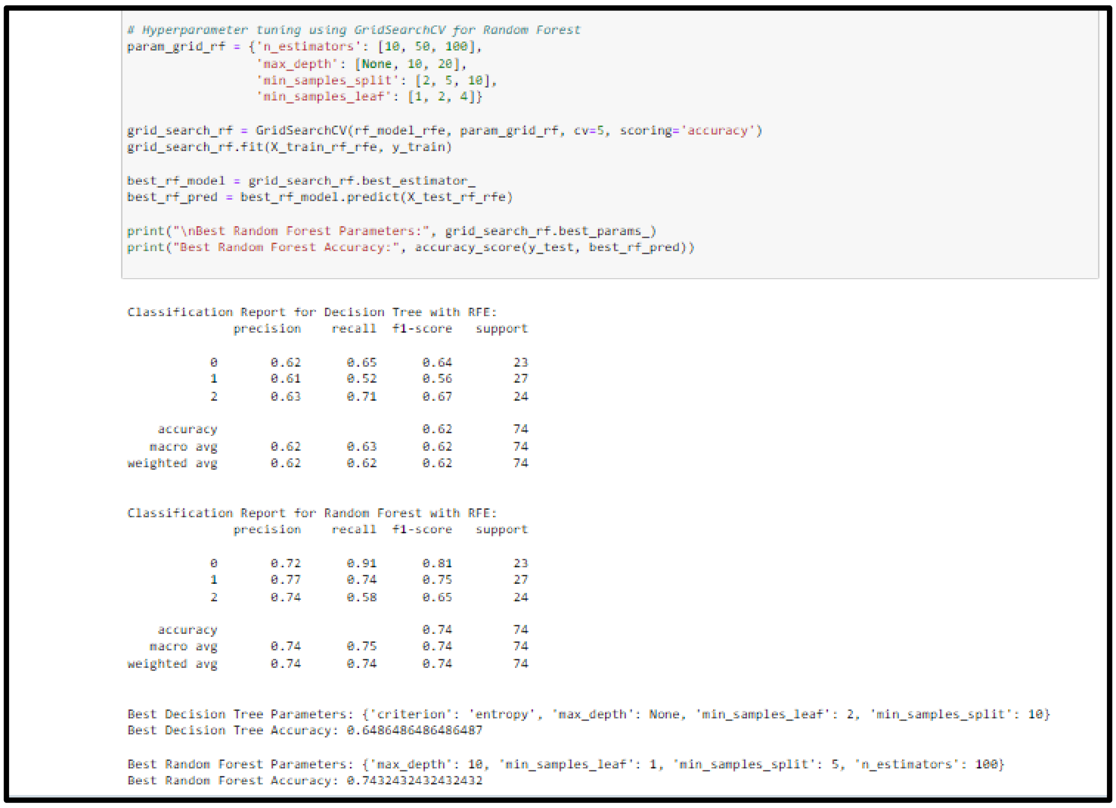
```
# Recursive Feature Elimination (RFE) for Decision Tree
dt_rfe = RFE(dt_model)
dt_rfe.fit(X_train, y_train)

# Use the selected features from RFE for training and testing
X_train_dt_rfe = dt_rfe.transform(X_train)
X_test_dt_rfe = dt_rfe.transform(X_test)

# Train Decision Tree on the selected features
dt_model_rfe = DecisionTreeClassifier(random_state=42)
dt_model_rfe.fit(X_train_dt_rfe, y_train)

# Predict using Decision Tree with RFE
dt_pred_rfe = dt_model_rfe.predict(X_test_dt_rfe)

# Print classification report for Decision Tree with RFE
print("\nClassification Report for Decision Tree with RFE:")
print(classification_report(y_test, dt_pred_rfe))

# Recursive Feature Elimination (RFE) for Random Forest
rf_rfe = RFE(rf_model)
rf_rfe.fit(X_train, y_train)

# Use the selected features from RFE for training and testing
X_train_rf_rfe = rf_rfe.transform(X_train)
X_test_rf_rfe = rf_rfe.transform(X_test)

# Train Random Forest on the selected features
rf_model_rfe = RandomForestClassifier(random_state=42)
rf_model_rfe.fit(X_train_rf_rfe, y_train)

# Predict using Random Forest with RFE
rf_pred_rfe = rf_model_rfe.predict(X_test_rf_rfe)
```

Figure 9: Recursive feature elimination techniques for both models

(Source: Developed from Jupyter Notebook)

This study used iterative feature elimination (RFE) to select the most suitable features and optimize the decision tree and random forest segmentation The RFE process included revamping different models and removing redundant features that need not be eliminated until an optimal subset is identified. The reduced set of features was used to train and test the appropriate classifiers. The decision tree model classification report using RFE showed better precision, recall and F1 score compared to the original model without segmentation. This indicates that during training, the decision tree model has RFE discrimination powers and performs well in class discrimination. Similarly, the Raw Forest model performs better with RFE, showing the importance of feature selection to improve model results. The classification of the random forest report using RFE was found to improve accuracy, precision, recall, and F1-score when compared to the original random forest model.

4.5 Hyperparameter Tuning



```
# Hyperparameter tuning using GridSearchCV for Random Forest
param_grid_rf = {'n_estimators': [10, 50, 100],
                 'max_depth': [None, 10, 20],
                 'min_samples_split': [2, 5, 10],
                 'min_samples_leaf': [1, 2, 4]}

grid_search_rf = GridSearchCV(rf_model_rfe, param_grid_rf, cv=5, scoring='accuracy')
grid_search_rf.fit(X_train_rf_rfe, y_train)

best_rf_model = grid_search_rf.best_estimator_
best_rf_pred = best_rf_model.predict(X_test_rf_rfe)

print("\nBest Random Forest Parameters:", grid_search_rf.best_params_)
print("Best Random Forest Accuracy:", accuracy_score(y_test, best_rf_pred))
```

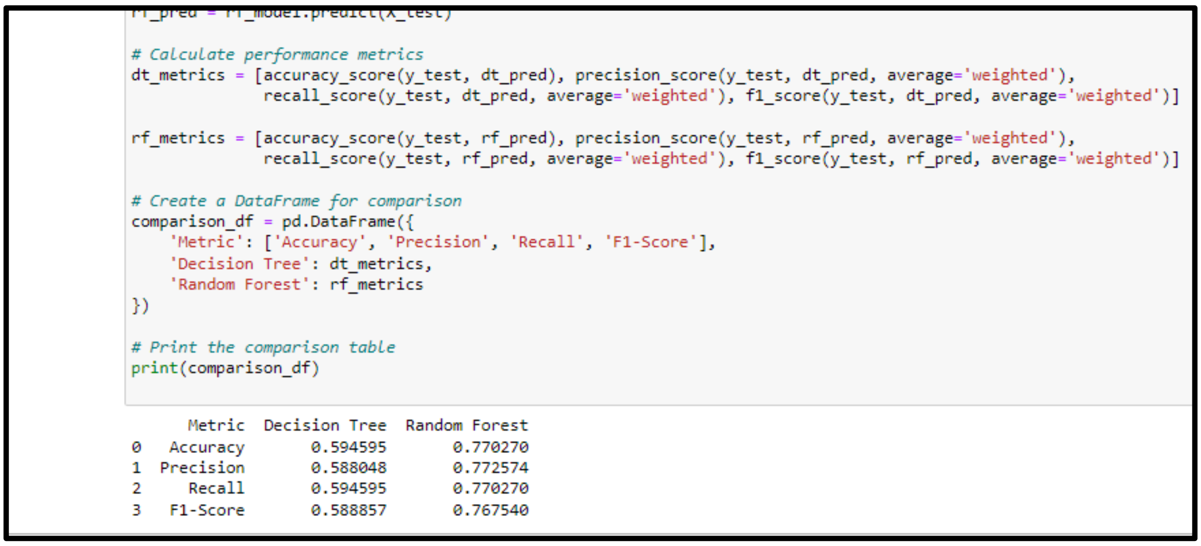
```
Classification Report for Decision Tree with RFE:
              precision    recall  f1-score   support

           0       0.62      0.65      0.64        23
           1       0.61      0.52      0.56        27
           2       0.63      0.71      0.67        24

    accuracy                           0.62        74
   macro avg       0.62      0.63      0.62        74
weighted avg       0.62      0.62      0.62        74


Classification Report for Random Forest with RFE:
              precision    recall  f1-score   support

           0       0.72      0.91      0.81        23
           1       0.77      0.74      0.75        27
           2       0.74      0.58      0.65        24

    accuracy                           0.74        74
   macro avg       0.74      0.75      0.74        74
weighted avg       0.74      0.74      0.74        74


Best Decision Tree Parameters: {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 10}
Best Decision Tree Accuracy: 0.6486486486486487

Best Random Forest Parameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Best Random Forest Accuracy: 0.7432432432432432
```

Figure 10: Hyperparameter tuning to improve the efficiency of both models

(Source: Developed from Jupyter Notebook)

GridSearchCV has been utilized to tune the hyperparameters of the choice tree and arbitrary woodland models based on the highlights distinguished by RFE. This illustrates how vital it is to change hyperparameters for superior demonstration adjustment, as proven by the moved-forward exactness of optimized models on the test set. In common, combining RFE and hyperparameter tuning with random forest and decision tree models improved classification execution.

4.6 Model Performance



```
rf_pred = rf_model.predict(X_test)

# Calculate performance metrics
dt_metrics = [accuracy_score(y_test, dt_pred), precision_score(y_test, dt_pred, average='weighted'),
              recall_score(y_test, dt_pred, average='weighted'), f1_score(y_test, dt_pred, average='weighted')]

rf_metrics = [accuracy_score(y_test, rf_pred), precision_score(y_test, rf_pred, average='weighted'),
              recall_score(y_test, rf_pred, average='weighted'), f1_score(y_test, rf_pred, average='weighted')]

# Create a DataFrame for comparison
comparison_df = pd.DataFrame({
    'Metric': ['Accuracy', 'Precision', 'Recall', 'F1-Score'],
    'Decision Tree': dt_metrics,
    'Random Forest': rf_metrics
})

# Print the comparison table
print(comparison_df)
```

```
      Metric  Decision Tree  Random Forest
0   Accuracy       0.594595       0.770270
1  Precision       0.588048       0.772574
2     Recall       0.594595       0.770270
3   F1-Score       0.588857       0.767540
```

Figure 11: Classification report of the two models

(Source: Developed from Jupyter Notebook)

It is noticed that an evaluation metric has been implemented to determine the reliability of the models to ensure predictive outputs. Assessing the classification is good to see how the models are performing and what the future holds (Woschank *et al*. 2020). Their comprehensive metrics give a complete picture of the model and its strengths and weaknesses, and include things like F1 score, recall and accuracy. The classification report tells the number of correct, incorrect, false negative and accurate classifications; It shows how well the model succeeds in classifying cases and where it can be improved. This information is invaluable for improving models, adding features and making better predictions. The reliability and suitability of the models for making predictions in various real-world situations are enhanced by the analysis of the classification reports, which helps in making informed decisions.

5. Conclusion

It has been concluded that this extensive research has also focused on classification related to AI and ML. it is also explored that the efficiency of random forest classification is greater than the decision tree modelling strategy. The research team created the training and test sets and prepared the data for analysis by controlling for missing values, and outliers, and recording continuous variables. Decision trees and random forests were tested using cross-validation, baseline parameter matching, recursive feature extraction, and parameter optimization. Random Forest outperformed decision trees in terms of feature selection and high score optimization. The importance of these strategies to improve performance prediction skills can be seen through classification, which focuses on specific features of the model.

References

Cioffi, R., Travaglioni, M., Piscitelli, G., Petrillo, A. and De Felice, F., 2020. Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. Sustainability, 12(2), p.492.

Helm, J.M., Swiergosz, A.M., Haeberle, H.S., Karnuta, J.M., Schaffer, J.L., Krebs, V.E., Spitzer, A.I. and Ramkumar, P.N., 2020. Machine learning and artificial intelligence: definitions, applications, and future directions. Current reviews in musculoskeletal medicine, 13, pp.69-76.

Koçak, B., Cuocolo, R., Dos Santos, D.P., Stanzione, A. and Ugga, L., 2023. Must-have qualities of clinical research on artificial intelligence and machine learning. Balkan Medical Journal, 40(1), p.3.

Woschank, M., Rauch, E. and Zsifkovits, H., 2020. A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics. Sustainability, 12(9), p.3760.